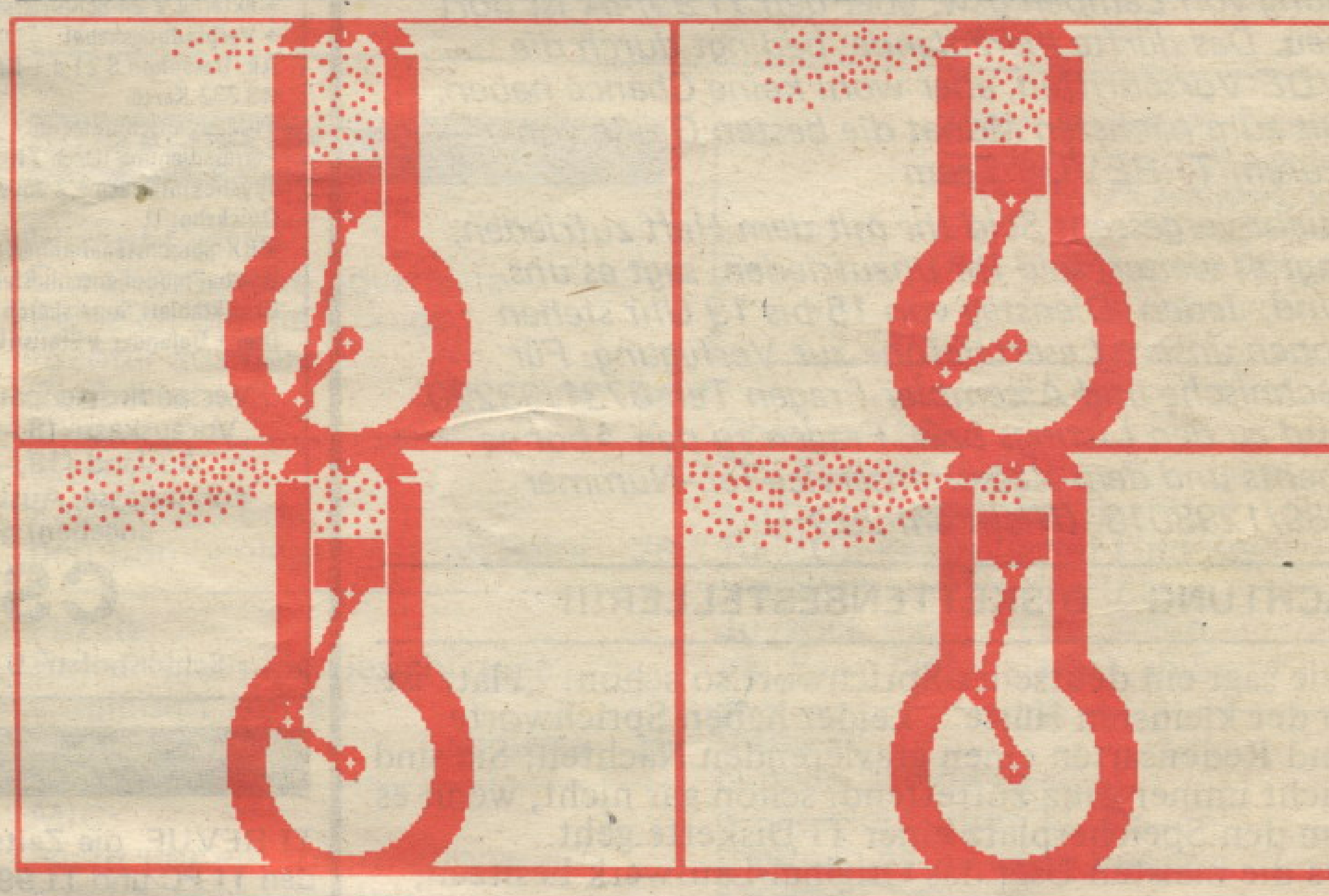


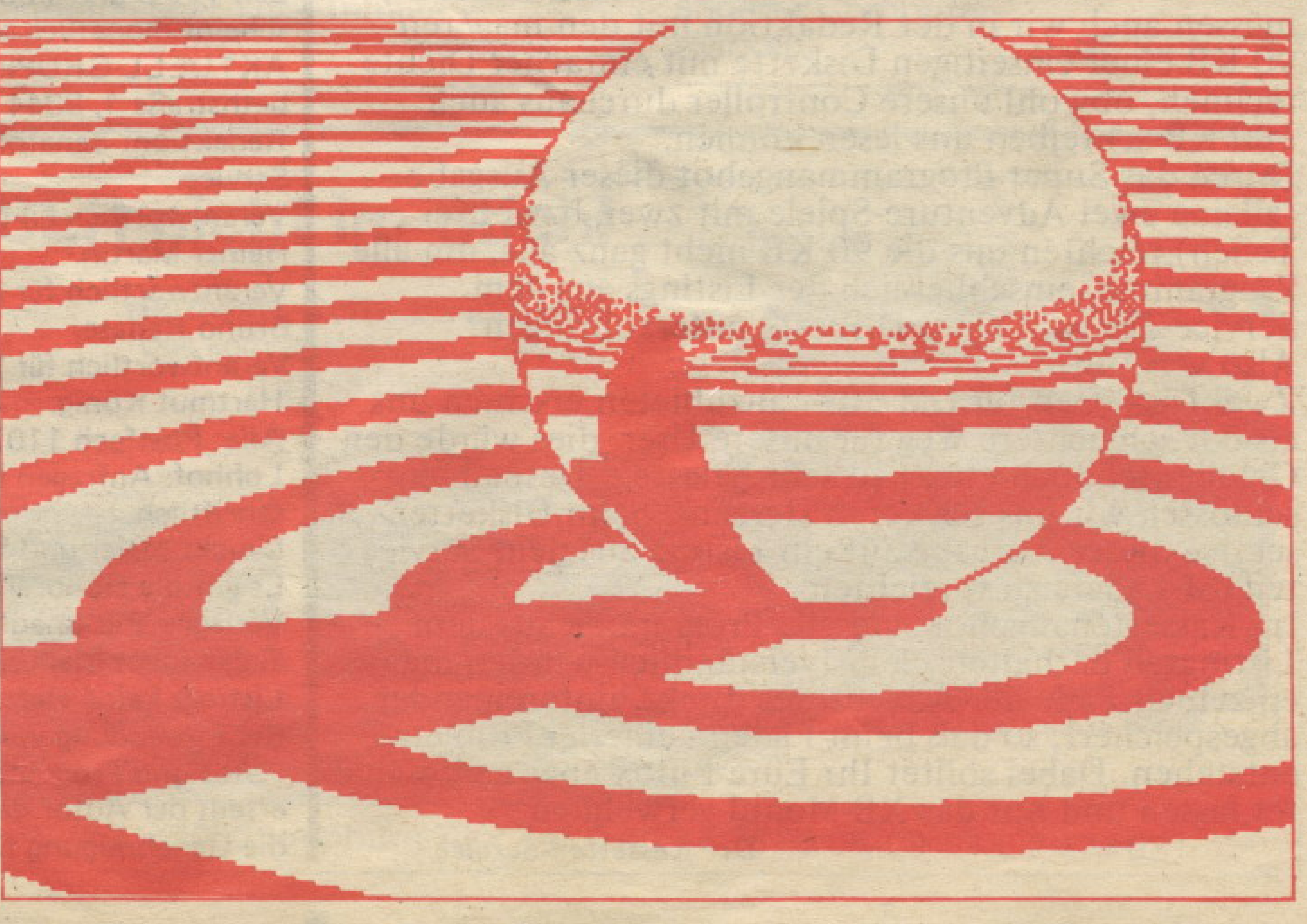
REVUE

Das Magazin für TI 99-4A

Nr. 8/86-August -DM 5,50 / ÖS 46 / SRF 5,50

BAUEN SIE SICH DOCH IHREN PLOTTER SELBST (II)





Grüß Gott - Guten Tag

Der neue „Computer" aus den USA scheint nun doch langsam Formen anzunehmen. Fest steht mittlerweile, daß es dieses Ding nur als Karte für die P-Box geben wird. Daran angeschlossen wird eine „normale" IBM-Tastatur. Die restlichen Daten sind inzwischen bekannt. Die bisherigen Module sollen alls, in ausgelesener Form laufen. Assemblerprogramme sollen zu 95% kompatibel sein. Die ersten Muster des Gerätes sollen schon ausgeliefert sein. Soweit die „offiziellen" Meldungen darüber aus den USA. Inoffiziellen Meldungen zufolge, d.h. von Messebesuchern, die das Gerät schon gesehen haben, soll es aber auch noch eine ganze Menge Probleme mit der Kompatibilität zu bisherigen Programmen geben.
Genug der Gerüchte, kommen wir zu bekannten Neuheiten: In den USA sind einige 512 KByte RAM-Erweiterungen neu auf den Markt gekommen. Auch ein Steuerungssystem zur Fernbedienung von Lampen usw. über den TI 99/4A ist dort neu. Das dürfte hierzulande, bedingt durch die VDE-Vorschriften, aber wohl keine Chance haben.
Bis zum nächsten Monat die besten Grüße von Eurem TI-REVUE-Team

Nicht vergessen: Seid Ihr mit dem Heft zufrieden, sagt es weiter, seid Ihr unzufrieden, sagt es uns.
Und: Jeden Dienstag von 15 bis 19 Uhr stehen Ihnen unsere Lesertelefone zur Verfügung. Für technische und Assembler-Fragen Tel. 0731/33220 und zu den Listings bzw. Fragen zu den Abonnements und dem Kassettenservice Tel.-Nummer 089/1298013. (Nicht am 29.7.!)

ACHTUNG – DISKETTENBESTELLER!!!

Wie sagt ein deutsches Sprichwort so schön: „Platz ist in der kleinsten Hütte". Leider haben Sprichworte und Redensarten einen gravierenden Nachteil: Sie sind nicht immer ganz zutreffend, schon gar nicht, wenn es um den Speicherplatz einer TI-Diskette geht.
Da die meisten User das Original-Laufwerk besitzen, müssen auch wir in der Redaktion mit den mageren 90 KB einer einseitigen Diskette mit einfacher Dichte rechnen, obwohl unsere Controller durchaus auch 180 KB schreiben uns lesen können.
Durch das Super-Programmangebot dieser Ausgabe (alleine zwei Adventure-Spiele mit zwei, bzw. drei Teilen) reichten uns die 90 KB nicht ganz aus, um alle Programme, einschließlich der Listings aus dem Service-Teil, auf einer Diskette unterzubringen?
Also was tun?
Zwei Disketten für DM 50,– anzubieten erschien uns als der schlechtere Weg für unsere User, dies würde den Geldbeutel doch etwas zu sehr belasten. Deshalb entschlossen wir uns schweren Herzens, beim Diskettenservice dieser Ausgabe auf ein Listing aus dem Serviceteil (M-Lader) zu verzichten.
Im Kassettenangebot sind alle Programme aus dem Listingteil enthalten. Selbstverständlich wurden die einzelnen Teile der Adventures direkt hintereinander abgespeichert, so daß beim Laden keinerlei Probleme entstehen. Dabei solltet Ihr Eure P-Box aber ausgeschaltet lassen und nur das XB-Modul verwenden.

Der Kassetten-Service

IMPRESSUM

TI-REVUE, die Zeitschrift für den TI PC und TI 99/4A erscheint monatlich in der AKTUELL-GRUPPE, Elisabethstraße 1, 8044 Lohhof.
Redaktion: Senator-Presse-Service.
Verantwortlich für den Inhalt: Heiner Martin.
Verantwortlich für Anzeigen: Bruno Redase.
Verantwortlich für Listings: Hartmut König.
Alle: Postfach 1107, 8044 Lohhof. Anfragen bitte nur schriftlich.
Druck: Maier und Söhne
Es gilt die Honorarliste des Verlages. Für unaufgefordert eingesandte Manuskripte und Listings keine Haftung.
Bei Einsendung von Texten, Fotos und Programmträgern erteilt der Autor dem Verlag die Genehmigung für einen einmaligen Abdruck sowie die Aufnahme in den Programm-Service nach den **Verlags-Sätzen und überträgt dem Verlag das Copyright!**
Alle in dieser Zeitschrift veröffentlichten Beitrage sind urheberrechtlich geschützt. Jedwede Verwertung ist untersagt. Nachdruck nur mit ausdrücklicher schriftlicher Zustimmung des Verlages.
Namentlich gezeichnete Artikel geben nicht unbedingt die Meinung der Redaktion wieder.
Kein Anspruch auf Lieferung bei Ausfall durch höhere Gewalt.
Gerichtsstand: München
Geschäftsführer: Werner E. Seibt
Abo- und Kassetten-Service: Henny Rose Seibt

INHALT TI-REVUE NR. 8/86

Bisher konnte die Tastatur nicht abgefragt werden, TI-REVUE schildert, wie es möglich ist. Ab Seite 17

ASSEMBLERPROGRAMME IM PROGRAMM IMAGE FORMAT EDITIEREN

Texas Instruments brachte, noch kurz vor dem Produktionsstop des TI-99/4A, die Programmiersprache TI-Forth auf den Markt. Können Sie mir sagen, wo man diese kaufen kann und worin der Unterschied zu dem ebenfalls erhältlichen TEX-Forth besteht? Weiterhin möchte ich Sie bitten, mir mitzuteilen, ob es eine Möglichkeit gibt, Assemblerprogramme, die im Programm Image Format abgespeichert sind, zu editieren und zu ändern?
Roland Kurz,
Nürnberg

Texas Instruments hat das unter dem Namen TI-Forth veröffentlichte Forth nicht mehr selbst herausgebracht. Nach der Einstellung der Produktion des Homecomputers verkaufte TI die Rechte an die Firma Texcomp. Diese Firma brachte dieses dann unter dem Namen TEX-Forth auf den Markt. Käuflich ist es hier bei einigen Händlern, die auch in der TI-REVUE annoncieren. Selbstverständlich ist es auch möglich, Assemblerprogramme, die im Programm Image Format stehen, zu editieren. Der einzige Hasenfuß liegt bei einem entsprechenden Programm, das dabei unumgänglich ist. In den USA gibt es von der Fa. Millers Graphics jetzt neu solch ein Programm. Bis dieses jedoch nach Deutschland gelangt, wird wohl noch einige Zeit vergehen.

NÜTZLICHE DISKETTENHINWEISE

Ich möchte Ihnen als erstes für den Beitrag 'Disketten unters Hemd geschaut' aus Heft 6/86 danken. Es kommt jedoch der Verdacht auf, daß dieser Artikel eine Übersetzung des Bediener Manuals des Disk-Fixer Moduls ist. Dies ist jedoch nicht der Hauptgrund meines Schreibens. Vielmehr betrifft mein Schreiben die Lücken, die der Beitrag ließ. Es sind keine gravierenden Mängel, zumal in der Kürze des Beitrages natürlich nicht alle Aspekte des Diskettenbetriebssystem behandelt werden konnten.
Nun die einzelnen Punkte: Sektor 0: Die Bedeutung des Strings 'DSK' ist folgendermaßen: Diese drei Bytes werden nach dem Initialisieren zusammen mit dem Sektor 0, dessen Bitmap durch den Verify-Vorgang aktualisiert wurde, auf die Diskette geschrieben. Sie geben die Information, ob eine Diskette überhaupt initialisiert ist. Verändert man eines dieser Bytes, so wird der Error 'Diskette nicht initialisiert' ausgegeben.
Sektor 2–33: Es ist nur teilweise richtig, daß diese Sektoren nur für den Diskettenkatalog (den File-Descriptor Sektoren oder auch FDS) reserviert sind. Sind alle weiteren Sektoren der Diskette belegt, so werden diese ebenfalls zur Datenspeicherung verwendet. Benötigt der Computer, nachdem diese FDS belegt sind, noch weitere FDS, so verwendet er den Sektor, der direkt vor dem neu abgespeicherten File steht. Der entsprechende Sektor wird zwar im Sektor 1 erwähnt, jedoch geht er meiner Meinung nach in der Block-Link unter.
Bei einigen Beispielen wurde die TI-Unart aufgegriffen. Die Bits 2, 4, 6 sind immer Null. Das Bit 0 gibt an, ob es sich um einen Datenfile (das Bit ist gleich 0) oder um einen Program-File (das Bit ist gesetzt), handelt. Die Bytes >12 und >13 gehören zusammen, da sonst die Belegung eines Files auf maximal >FF Sektoren begrenzt würde. Als einziger wichtiger Punkt ist dabei zu sagen, daß diese vertauscht werden müssen, um die Länge des Files herauszufinden.
Bei der Block-Link (auch Cluster-)Map ist zu sagen, daß der Sektor-Offset von Block zu Block fortlaufend ist.
Christopher Winter,
Obertshausen

Vielen Dank für Ihre hilfreichen Tips zur Verwendung eines Disketten Editors. Es ist jedoch noch zu bemerken, daß es inzwischen eine verhältnismäßig große Anzahl von solchen Hilfsprogrammen gibt. Das Disk-Fixer-Modul ist bei weitem nicht das einzige. Ihr Brief stellt jedoch einen hilfreichen Nachtrag zu dem von Ihnen genannten Beitrag dar.

SCREEN-OFFSET IN X-BASIC

Ich versuche ein Assemblerprogramm, das ich für den Editor/Assembler geschrieben habe, für X-Basic umzuschreiben. Dabei habe ich bei einer Unterroutine größere Probleme: Um die Anleitung in X-Basic auszugeben, benötigt man, um ein Zeichen auf den Bildschirm zu bringen, einen Screen-Offset von >60. Mir gelingt es jedoch nicht, den Offset zu den Zeichen dazu zu addieren. Können Sie mir sagen, wie diese Unterroutine in Assembler aussieht?
Mike Rohrmoser,
Essen

Zu Ihrem Problem ist zu bemerken, daß Sie mit den schon integrierten Unterprogrammen VMBW und VMBR nicht sehr weit kommen. Eine Möglichkeit wäre VSBW, welche jedoch sehr langsam ist. Also bleibt dann nur die folgende Lösung, da sie in der Geschwindigkeit am besten abschneidet:

```
L LI RO, >10 * Bildschirmadresse laden, wenn
0, dann CLR R0 L I R1,
HELP1 * Pointer zum
Text laden
LI R2,778 * Anzahl der
zu übertragenden Bytes
BL %VBTW * Sprung zum
Unterprogramm
 * Hier weiteres Programm
XBTW  ORI R0,>4000
      SWPB R0
      MOVB R0,@>8C02
      SWPB R0
      MOVB R0,@
      >8C02 * VDP-
      Adresse
geschrieben
XBTWL MOVB *R1+,R0
      AI R0,>6000 *
Offset in Byte addieren
      MOVB R0,@
      >8C00
      DEC R2
      JNE XBTWL
      RT
```

Dies ist die ganze Hexerei.

GPLLNK VOM TITELBILD AUS AUFRUFEN

Wir sind ein kleiner TI-Club, der sich hauptsächlich mit der Erstellung neuer Module beschäftigt. Unser größtes Problem ist das Aufrufen der Unterroutine GPLLNK vom Titelbild aus. Wir haben die Unterroutinen aus dem E/A-Modul in die Speichererweiterung ausgelesen, so daß ein Aufruf über REF möglich ist. Alle Unterprogramme, wie z.B. VMBW und DSRLNK, laufen einwandfrei. Allein GPLLNK bereitet Schwierigkeiten. Es wäre nett, wenn Sie uns helfen könnten, unser Problem zu lösen.
TI/99 Club,
Peter Vorwerk,
Pinneberg

Die Unterroutine GPLLNK ist sehr stark von dem Modul abhängig, für welches sie geschrieben wurde. Sowie im E/A-Modul als auch im Minimem besteht das GPLLNK nicht nur aus einem reinen Assemblerteil, sondern beinhaltet noch GPL-Routinen. Das X-Basic

GPLLNK, das im Assembler Sonderheft verwendet wurde, trickst ein wenig. Es verwendet als Rücksprungadresse von GPL den eigentlichen Einsprungspointer des CALL LINK Aufrufes, nur wird dieser Pointer 'umgebogen'. Uns ist zwar eine reine ROM Version bekannt, jedoch können wir diese aus urheberrechtlichen Gründen nicht veröffentlichen. Einen Tip können wir Ihnen aber geben: Diese Routine ist im deutschen FORTH-Modul programmiert.

MEHR SPEICHERPLATZ ALS 24 K ANSCHLIESSBAR

Ich habe mich entschlossen, Ihnen endlich alle meine angesammelten Fragen zum TI-99 zu stellen. Ich habe sie Ihnen in Form einer Liste zusammengefaßt:

1. Gibt es für den TI ein Programm bzw. Modul, mit dem ich richtige Bilder erstellen kann?
2. Gibt es die Möglichkeit, daß dem TI mehr als nur 24 K (wie bei der 32 K-Erweiterung) zur Verfügung stehen?
3. Ist es normal, daß nach längerem Einschalten mein TI sich am Modulschacht stark erhitzt?
4. Welchen Befehl muß ich eingeben, um den Bildschirm im laufenden Programm ausdrucken zu lassen?
5. Ist es normal, daß ein X-Basic Programm nach dem Abschalten der P-Box gelöscht ist?
6. Ist es möglich, fremde Basic-Dialekte dem TI anzupassen, wenn ja, wie?

Thomas Nowak, Berlin

Wir werden versuchen, alle Ihre Fragen zu beantworten:
1. Ein Grafikprogramm mit den von Ihnen genannten Eigenschaften, den TI Artist, hatten wir gerade in der TI-REVUE 5/86 vorgestellt.
2. Der TMS9900 kann direkt nur 24 K adressieren. Mit besonderen Erweiterungen sind aber mehr durchaus möglich.
3. Die Erwärmung am Modulport ist durchaus normal, da sich darunter der Spannungsregler befindet. Dieser erwärmt nach einiger Betriebszeit die schwarze Fläche.
4. Wenn Sie einen Bildschirmausdruck erstellen wollen, so benötigen Sie dazu eine Hardcopyroutine. Diese sind aber von Drucker zu Drucker verschieden. Einige Routinen zu verschiedenen Druckertypen haben wir in der TI-REVUE schon veröffentlicht.
5. Nach dem Abschalten der P-Box muß ein X-Basic Programm verloren sein, da es sich in der RAM-Erweiterung befindet. Schaltet man den Strom der P-Box ab, so bekommt die RAM-Erweiterung ebenfalls keinen Strom mehr. Dadurch wird das Programm gelöscht.
6. Basic-Programme anderer Dialekte müssen Sie wohl oder übel 'übersetzen'. Das heißt, Sie müssen alle Befehle, die der TI nicht kennt, umschreiben.

X-BASIC DEFEKT

Ich wollte ein Programm zum Erstellen von Texten in 32 Zeilen zu je 28 Zeichen schreiben. Doch schon beim Eingabeprogramm bin ich gescheitert. Die ersten 24 Zeilen einzugeben, funktioniert fehlerlos. Will man jedoch die letzten 8 Zeilen eingeben, so nimmt der TI nur noch die Cursor-Tasten an. Was ist da falsch? Liegt es an meinem X-Basic oder am CPS99? Die Programmzeilen sehen folgendermaßen aus:

```
140 FOR D=1 TO 8 :: ACCEPT
AT (D,1): A$(D+24) ::
NEXT D
```

Die Abspeicherroutine scheint zu funktionieren, auch die letzten 8 AS (D)'s. Doch beim Abspeichern schneidet der TI immer an beliebigen Stellen Wörter ab.
Woran liegt das? Das Listing sieht folgendermaßen aus:

```
100 CALL CLEAR
110 DIM A$(32)
120 FOR D=1 TO 32
130 IF D  25 THEN
      Z=D ELSE Z=D-24
140 IF D=25 THEN CALL
      CLEAR
150 ACCEPT AT(Z,1)
      :A$(D)
160 NEXT D
170 DISPLAY AT(14,1):
      "DATEINAME:
      DSK1.SEITE"
180 ACCEPT AT(14,17)
      SIZE (-10):N$
190 OPEN
#1:"DSK1."NS , OUTPUT,
FIXED 30
200 FOR D=1 TO 32
210 PRINT #1:A$(D)
220 NEXT D
230 CLOSE #1
240 GOTO 100
```

Ich hoffe, Sie können mir bei meinem Problem weiterhelfen.

Joachim Venekens, Mayen

Zuerst zu Ihrem zweiten Problem, das Sie ansprachen: Verwenden Sie bei Ihrer Datei die Atribute INTERNAL, OUTPUT, FIXED 30. Ohne das INTERNAL liest der Computer beim Wiedereinlesen nur bis zu einem Trennungszeichen, z.B. Komma. Der Rest wird abgeschnitten.
Ihr erstes Problem ist etwas heikel. Um Sie im Voraus zu beruhigen: An Ihren Geräten stimmt alles. Im ACCEPT AT von TI scheint offensichtlich ein Fehler vorzuliegen: Verwendet man als Zielvariable ein Datenfeld, in Ihrem Fall A$(D), so treten zwei Fälle auf: Das ACCEPT AT funktioniert fehlerlos, solange man absolute Werte der Variablen zuteilt, z.B. A$ (3) oder man eine reine Variable verwendet, z.B. A$(D).
Der zweite Fall ist der, bei dem der Fehler auftritt. Steht in den Klammern des Datenfeldes ein Term, bei Ihnen A$(D+24), so nimmt der Computer kein Zeichen mehr an. Gibt man nun aber über VALIDATE an, welche Zeichen er nehmen darf, so funktioniert alles scheinbar fehlerlos. Komisch wird es erst, wenn man mehr als 256 Zeichen eingibt. Das ACCEPT AT akzeptiert so viele Zeichen, wie man will. In dem String werden aber nur die ersten 256 gespeichert. Gibt man mehr Zeichen ein als auf den Bildschirm passen und drückt dann ENTER, so kann ein Systemabsturz die Folge sein.

UNBEKANNTE UNTERPROGRAMME IM DISKCONTROLLER

Seit Sie in der Juni-Ausgabe der TI-REVUE auf die Unterprogramme des Diskcontrollers eingegangen sind, habe ich einige Fragen an Sie. Wie ich feststellte, sind im EPROM des Diskcontrollers noch weitere Unterprogramme enthalten. Die Programme '0110' und '0116' sind mir ein Begriff. Dennoch lassen mich die Unterprogramme '0111 bis '0115' im Unklaren.

Michael Kovar, Berlin

Der Disk-Controller ist von uns noch nicht vollständig analysiert worden, so daß wir Ihnen nur grobe Anhaltspunkte geben können.
'0111' dient zum 'roh'-formatieren. Die Sektoren 0 und 1 werden dabei nicht geschrieben. Die weiteren Unterprogramme enthalten das Ändern von Dateinamen und Löschen von Dateien. Weiter sind zwei Unterprogramme vorhanden, die das blockweise Lesen und Speichern von Dateien erlauben. Wir werden aber darauf noch in der TI-REVUE zurückkommen.

USER DEFINED INTER-RUPT

In der letzten Zeit erreichten mich einige Anfragen nach dem User-defined Interrupt des TI 99/4A. Mit dieser Funktion scheint es einige Schwierigkeiten zu geben, so daß wir uns heute einmal näher damit beschäftigen wollen.

Im Gegensatz zu einigen anderen Prozessoren, die auch ein softwaremäßiges Aufrufen eines Interruptes erlauben (was ja aber eigentlich nur eine besondere Form eines Unterprogrammaufrufes ist), gibt es beim TMS 9900 nur den Hardware-Interrupt, d.h., eine bestimmte Steuerleitung sagt dem Prozessor, daß er, sobald möglich, die Ausführung des normalen Programms unterbricht und eine besondere Interrupt-Routine ausführt.
Beim TI 99/4A wird ein Interrupt nun von verschiedenen Komponenten ausgelöst. Einmal benutzt die Kassettenroutine Interrupts zur Steuerung der Baud-Rate, dann können externe Erweiterungen (z.B. Druckerschnittstellen) Interrupts auslösen und dann gibt der Video-Prozessor alle 50stel Sekunde einen Interrupt an den Prozessor. Die ersten beiden Möglichkeiten interessieren uns hier nicht, denn sie haben mit dem User-defined Interrupt nichts zu tun.
Erkennt der TMS 9900 einen Interrupt, führt er im Betriebssystem die dafür vorgesehene Routine aus (im Betriebssystem ab >0900). In dieser wird nun erst einmal geprüft, um welchen Interrupt es sich handelt. Wie gesagt, interessiert uns hier nur der VDP-Interrupt. Dabei wird nun das Interrupt-Flag Byte auf >83C2 geprüft und je nach dessen Wert werden die einzelnen Routinen für die automatische Spritebewegung, die Soundlistensteuerung und für die Abfrage der Quittaste ausgeführt. Anschließend wird noch der Screentimeout-Zähler erhöht und eine Kopie des VDP-Statusregisters auf >83D4 gelegt. Jetzt folgt das Interessante: Es wird die Speicherstelle >83C4 abgefragt. Ist der Inhalt 0, geschieht gar nichts und der Interrupt wird beendet, ist diese ungleich 0, wird der Inhalt als Adresse einer auszuführenden Routine behandelt und diese über BL angesprungen. Übrigens erfolgt dies völlig unabhängig vom Wert des Interrupt-Flag Bytes, d.h., ein User-defined Interrupt kann darüber nicht abgeschaltet werden.
Um also während jedes Interruptes auch eine selbstbestimmte Interrupt-Routine auszuführen, müssen wir die Speicherstelle mit einem Pointer auf den Beginn unserer Routine legen. Dies geschieht aus dem Basic (bzw. natürlich Extended Basic) mit Hilfe eines einfachen Programms:

```
*Aufruf mit CALL LINK
("INTER")
        DEF   INTER
INTER   LI    R1,PNTR
        MOV   R1,@>83C4
        B     @>0070
```

Dabei brauchen wir, weil nur ein einziges Register benötigt wird, nicht extra einen eigenen Workspace zu bestimmen. PNTR ist dabei der Pointer zum Beginn der Routine.
Wenn die selbstgeschriebene Interruptroutine nicht mehr ausgeführt werden soll, wird der Pointer wieder mit folgendem Programm gelöscht:

```
*Aufruf mit CALL LINK
("CLRINT")
        DEF   CLRINT
CLRINT  CLR   @>83C4
              @>0070
```

Wie gesagt, das ist nicht der Rücksprung für die Interruptroutine, sondern es wird dann nicht mehr ausgeführt.
Jetzt brauchen wir aber zuerst noch eine kleine Routine, die dann bei jedem Interrupt ausgeführt werden soll. Beschreiben wir also den Bildschirm immer an einer ganz bestimmten Stelle mit einem "A". Die "User-defined Interrupt Routine" könnte damit so aussehen:

```
MYWS    BSS   32
PNTR    LWPI  MYWS
        LI    R0,30
        LI    R1,>A100 *Offset
        BLWP  @VSBW
        LWPI  >83E0
        RT
```

Beim Einsprung aus dem Betriebssystem ist der GPLWS (>83E0) geladen. Damit wir mehr Freiheiten in der Registerbenutzung haben, wird zuerst ein eigener Workspace geladen. Dann erfolgt in bekannter Weise das Schreiben des "A" auf den Bildschirm. Zum Rücksprung in die Interrupt-Routine des Betriebssystems müssen wir wieder GOLWS laden und dann, da ja der Einsprung aus dem Betriebssystem über BL (!) erfolgte, wieder mit RT, was ja gleich B *R11 ist, zurückgehen. Auf gar keinen Fall dürfen wir hier das B @>0070 verwenden. Werden diese Routinen zusammen assembliert (wegen des Labels PNTR notwendig) und dann von einem Basicprogramm aus aufgerufen, steht immer in der oberen rechten Ecke des Bildschirms ein "A". Im Basicprogramm sieht das dann wie folgt aus:

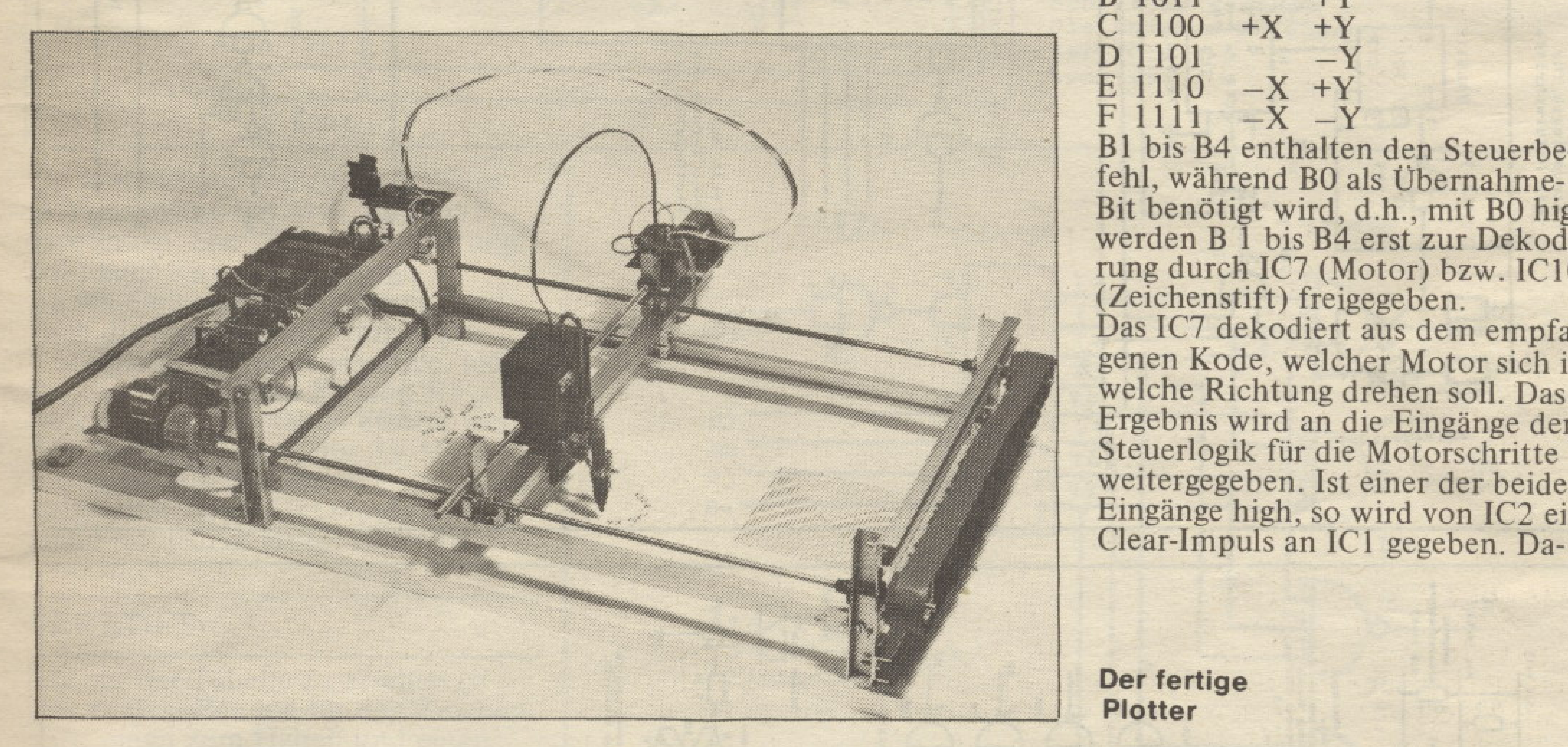
```
100 CALL LINK ("INTER")
    Weiteres Programm
500 CALL LINK ("CLRINT")
510 END
```

Das war dann eigentlich schon die ganze Hexerei. Rekapitulieren wir aber noch einmal: Das, was im Handbuch zum Editor/Assembler als User-defined Interrupt beschrieben ist, muß mit einem Pointer auf den Anfang der gewünschten Routine gelegt werden. Ist diese Speicherstelle 0, passiert aber gar nichts. Der Einsprung in die Routine erfolgt aus dem Betriebssystem mit BL, wobei der GPLWS geladen ist. Die Rückkehr aus der selbstgeschriebenen Unterroutine muß immer mit einem B *R11 bzw. RT erfolgen, wieder bei geladenem GPLWS.
Wer übrigens keinen eigenen Workspace für die Interruptroutine verwenden will, kann auch im GPLWS bleiben, es dürfen dann aber die Register R11, R13, R14 und R15 nicht geändert, also z.B. auch kein BLWP durchgeführt werden.
Der Interrupt wird, wie anfangs erwähnt, ja vom VDP 50mal in der Sekunde ausgelöst (bei amerikanischen Konsolen 60mal). Nun gibt es da aber einige Betriebsbedingungen, die den Interrupt abschalten. Dazu gehören einmal länger dauernde Zugriffe auf Peripheriegeräte wie z.B. den Diskcontroller. Auch sind viele Maschinenprogramme so aufgebaut, daß sie keinen Interrupt erlauben. Grundsätzlich kann hier gesagt werden, daß der User-defined Interrupt dann auch nicht ausgeführt wird, wenn Sprites auf dem Bildschirm stehen bleiben.
Zum Schluß darf der Hinweis nicht fehlen, daß man seine selbstgeschriebene Interrupt-Routine so kurz wie möglich halten sollte. Die Ausführung des Hauptprogramms wird sonst sehr langsam. Wenn nämlich diese z.B. länger als eine 50stel Sekunde dauert, wird die Ausführung des Hauptprogramms sogar gestoppt.

Heiner Martin

PLOTTER-MAL SELBST GEBAUT

Zu den reizvollsten Anwendungen unseres Computerhobbies gehört sicherlich das Plotten, das heißt, das programmgesteuerte Zeichnen von Funktionen, Kurven und Grafiken aller Art. Plotter für Heimcomputer sind relativ teuer, außerdem ist der Selbstbau viel interessanter. Mit dieser Anleitung wird beschrieben, wie man mit einfachen Mitteln einen Plotter bauen kann, der, gesteuert von dem TI 99/4A über ein CRU-Interface, ansehnliche Zeichnungen anfertigen kann. Im zweiten Teil dieses Beitrages geht es um die Steuerung.



Der fertige Plotter

Um eine exakte Bewegung des Stiftes zu erreichen, ist es notwendig, die Motoren so zu steuern, daß einem Steuerimpuls immer die gleiche Anzahl Umdrehungen des Motorankers bzw. der Gewindestangen zugeordnet ist. Um die Drehung der Motorwelle takten zu können, wird eine Lichtschranke an ein Getrieberad des Motorgetriebes angebaut. Das Getrieberad wird zur Hälfte mit lichtundurchlässigem

DIE STEUERUNG (Mechanik)

Papier beklebt. Auf diese Weise erhält man die Impulsfolge: aus – ein – aus für eine Umdrehung des Rades. Durch die weitere Untersetzung von 4:1 dreht sich die Gewindestange bei einer Taktfolge um eine Viertel-Umdrehung weiter.
Da dieser Vorgang für beide Motoren unabhängig voneinander gilt, ist für jeden Motor die Schaltung "Steuerlogik" einmal vorhanden.

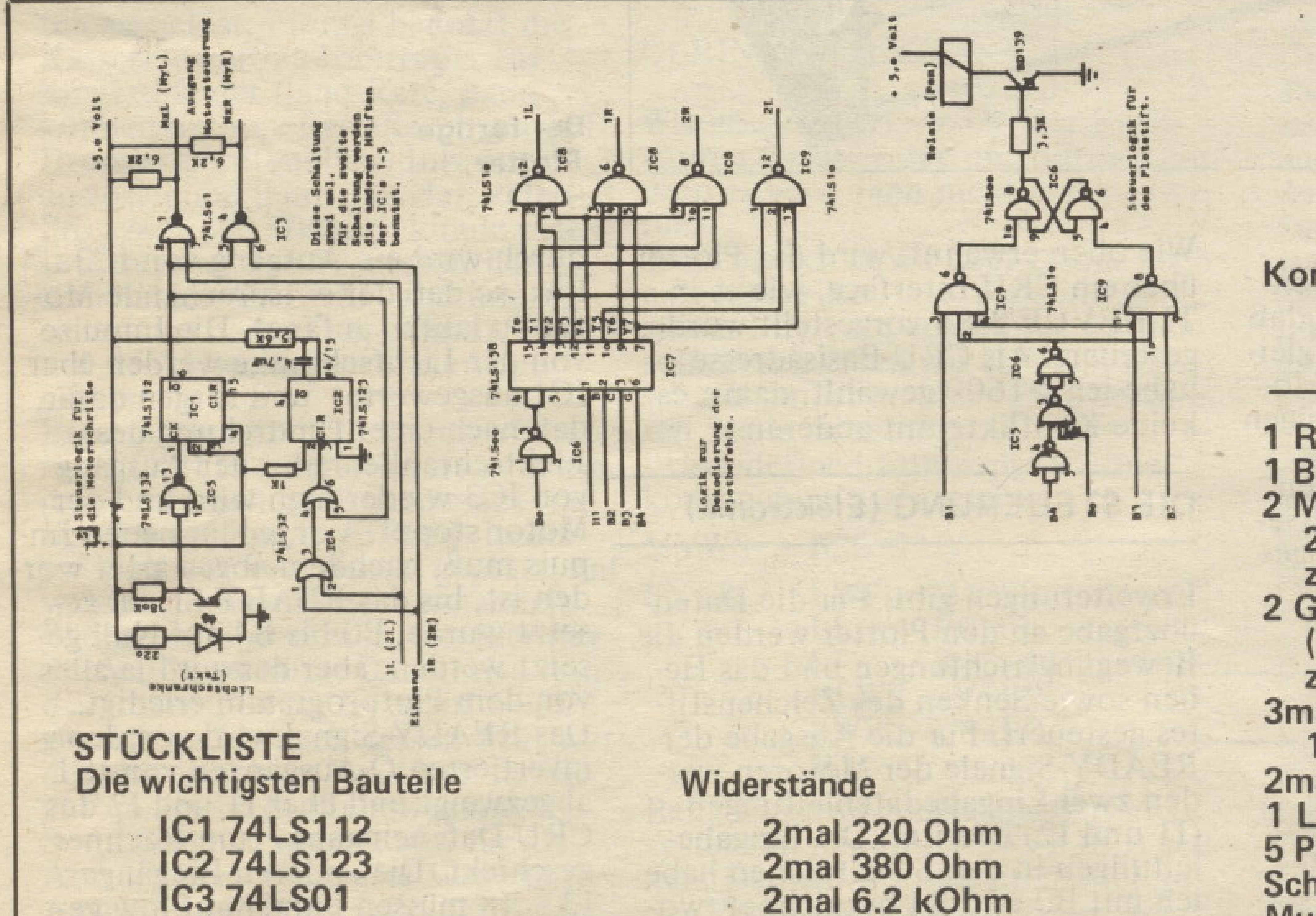
Wie oben erwähnt, wird der Plotter über ein CRU-Interface, wie es in TI-REVUE 2/84 vorgestellt wurde, gesteuert. Als CRU-Basisadresse habe ich >1600 gewählt, damit es keine Konflikte mit anderen

DIE STEUERUNG (Elektronik)

Erweiterungen gibt. Für die Datenübergabe an den Plotter werden die Bewegungsrichtungen und das Heben sowie Senken des Zeichenstiftes gesteuert. Für die Ausgabe der READY-Signale der Motoren werden zwei Eingabedatenleitungen (I1 und I2) benutzt. Die Eingabeleitungen in den Schaltplänen habe ich mit B0 und B4 bezeichnet, wobei B0 das niederwertigste Bit darstellt. Zur Steuerung des Plotters gilt folgende Code-Tabelle, die von der Steuerlogik entschlüsselt werden kann:

	B4 B1	
0	0000	Reset
1	0001	bis 4 0100 wird nicht benutzt
5	0101	Stift hoch
6	0110	Stift zeichnet
7	0111	frei
8	1000	+X
9	1001	–X
A	1010	+X –Y
B	1011	+Y
C	1100	+X +Y
D	1101	–Y
E	1110	–X +Y
F	1111	–X –Y

B1 bis B4 enthalten den Steuerbefehl, während B0 als Übernahme-Bit benötigt wird, d.h., mit B0 high werden B 1 bis B4 erst zur Dekodierung durch IC7 (Motor) bzw. IC10 (Zeichenstift) freigegeben.
Das IC7 dekodiert aus dem empfangenen Kode, welcher Motor sich in welche Richtung drehen soll. Das Ergebnis wird an die Eingänge der Steuerlogik für die Motorschritte weitergegeben. Ist einer der beiden Eingänge high, so wird von IC2 ein Clear-Impuls an IC1 gegeben. Dadurch wird ein Ausgang von IC3 low, so daß der entsprechende Motor zu laufen anfängt. Die Impulse von der Lichtschranke werden über IC1 ausgewertet und sorgen dafür, daß nach einer Umdrehung des Lichtschrankenrades der Ausgang von IC3 wieder high wird und der Motor stoppt. Vor jedem neuen Impuls muß, nachdem abgewartet worden ist, bis das READY-Signal gesetzt wurde, B0 bis B4 auf Null gesetzt werden; aber das wird ja alles von dem Plotprogramm erledigt.
Das READY-Signal wird von den invertierten Q-Ausgängen von IC1 abgezweigt und über I1 und I2 des CRU-Dateneingangs zum Rechner geschickt. Die anderen Eingänge I3 – I8 müssen immer auf low gelegt sein.
Zur Umsetzung des TTL-Signals für die Motorsteuerung habe ich eine Schaltung aus "Elektor 7/84 (64)" eingesetzt. Die Eingänge "A" und "B" dieser Schaltung werden mit den Ausgängen MxL und MxR für



STÜCKLISTE
Die wichtigsten Bauteile

- IC1 74LS112
- IC2 74LS123
- IC3 74LS01
- IC4 74LS32
- IC5 74LS132
- IC6 74LS00
- IC7 74LS138
- IC8 74LS10
- IC9 74LS10

Widerstände

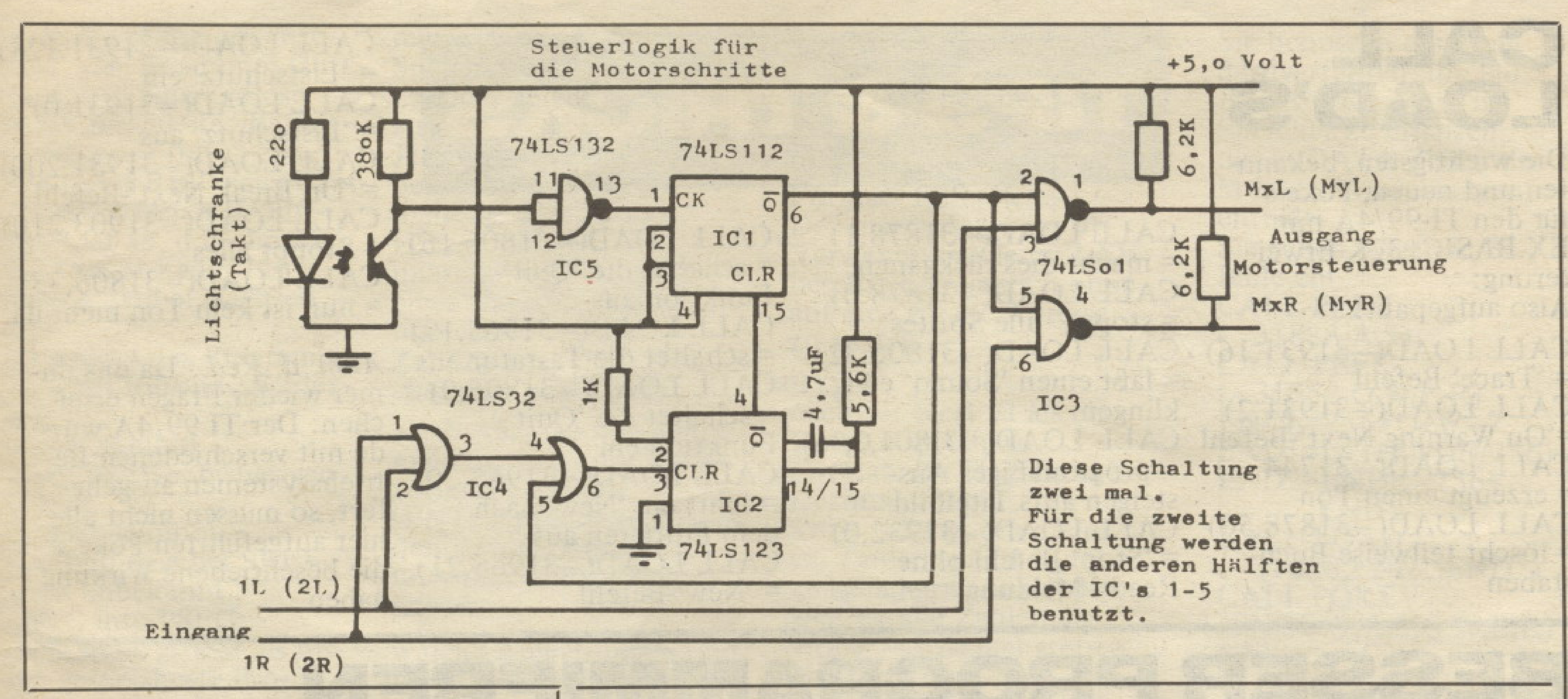
- 2mal 220 Ohm
- 2mal 380 Ohm
- 2mal 6.2 kOhm
- 2mal 5.6 kOhm
- 2mal 1.0 kOhm
- 1mal 3.3 kOhm
- 2mal 10 Ohm Watt

Kondensatoren

- 2mal 4.7 µF Elko
- 2mal 100µ F Keramik

1 Relais 5 V
1 BD139
2 Motoren (Mabuchi-Motor) 2.5V bis 4.5 V = z.B.: Völkner 0311308
2 Gabellichtschranken (NEC PS4001) z.B.: Völkner 0116680
3mal 1 m Alu U-Profilstäbe 16 mm mal 10 mm
2mal 1 m Gewindestangen M5
1 Lochrasterplatine
5 Platinenschienen
Schrauben M3 mal 10 mm
Muttern M3 und M5
Stecker und Flachbandkabel

Bauteile und Schaltbild der Motorsteuerung siehe 'Elektor', Heft 7/84



die X-Richtung und für den Motor in Y-Richtung mit MyL und MyR verbunden. Die "Elektor-Schaltung" wird auch zweimal benötigt.
Da ich die gesamte Schaltung, also auch die Motoren, mit nur einer 5 V-Spannungsquelle versorge, müssen die Widerstände R8 und R5 durch 1k5 Widerstände ersetzt werden. Zur Anpassung der 3V-Motoren an die 5 V-Versorgungsspannung habe ich je einen 10 Ohm-Widerstand mit jedem Motor in Reihe geschaltet.

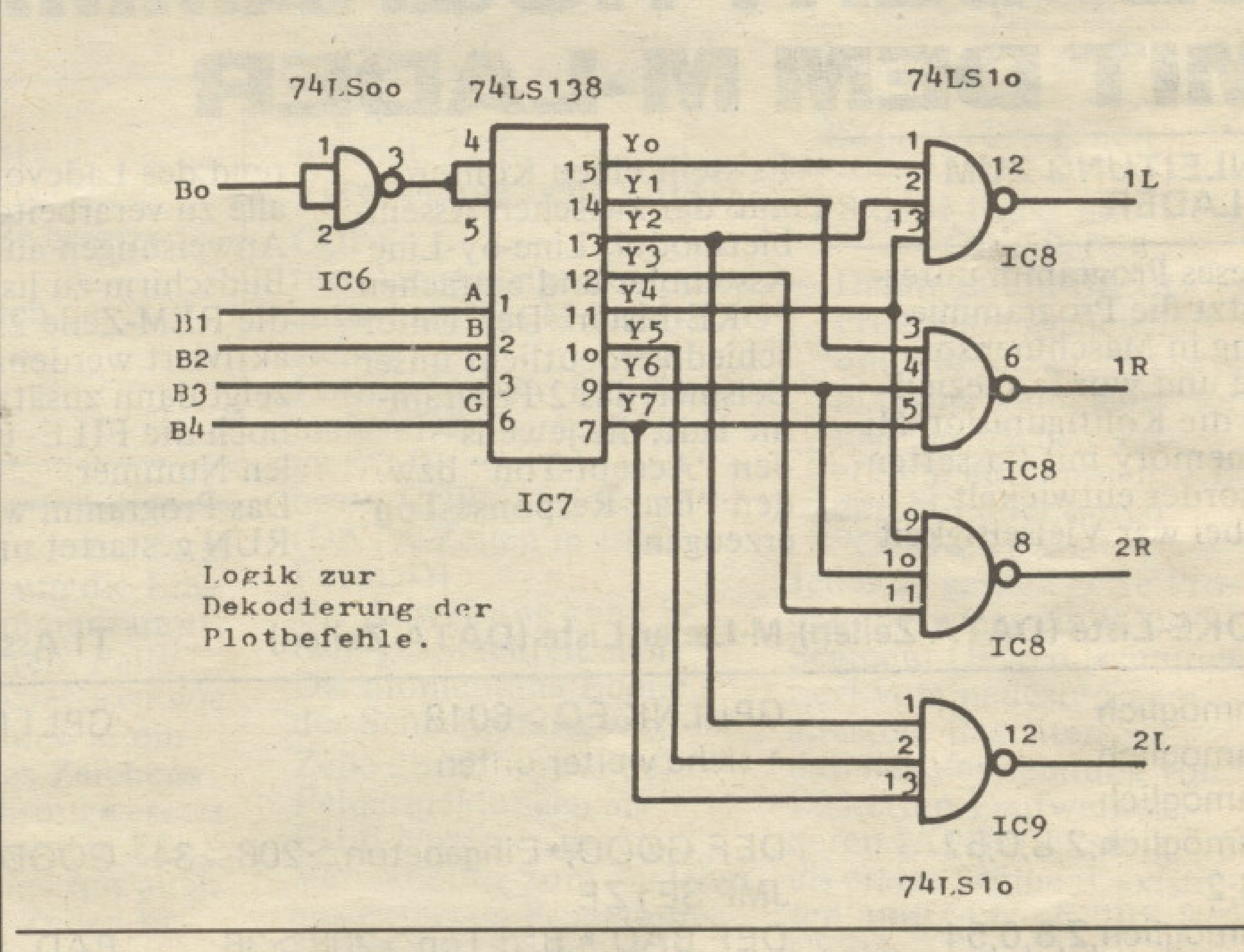
DER STIFT

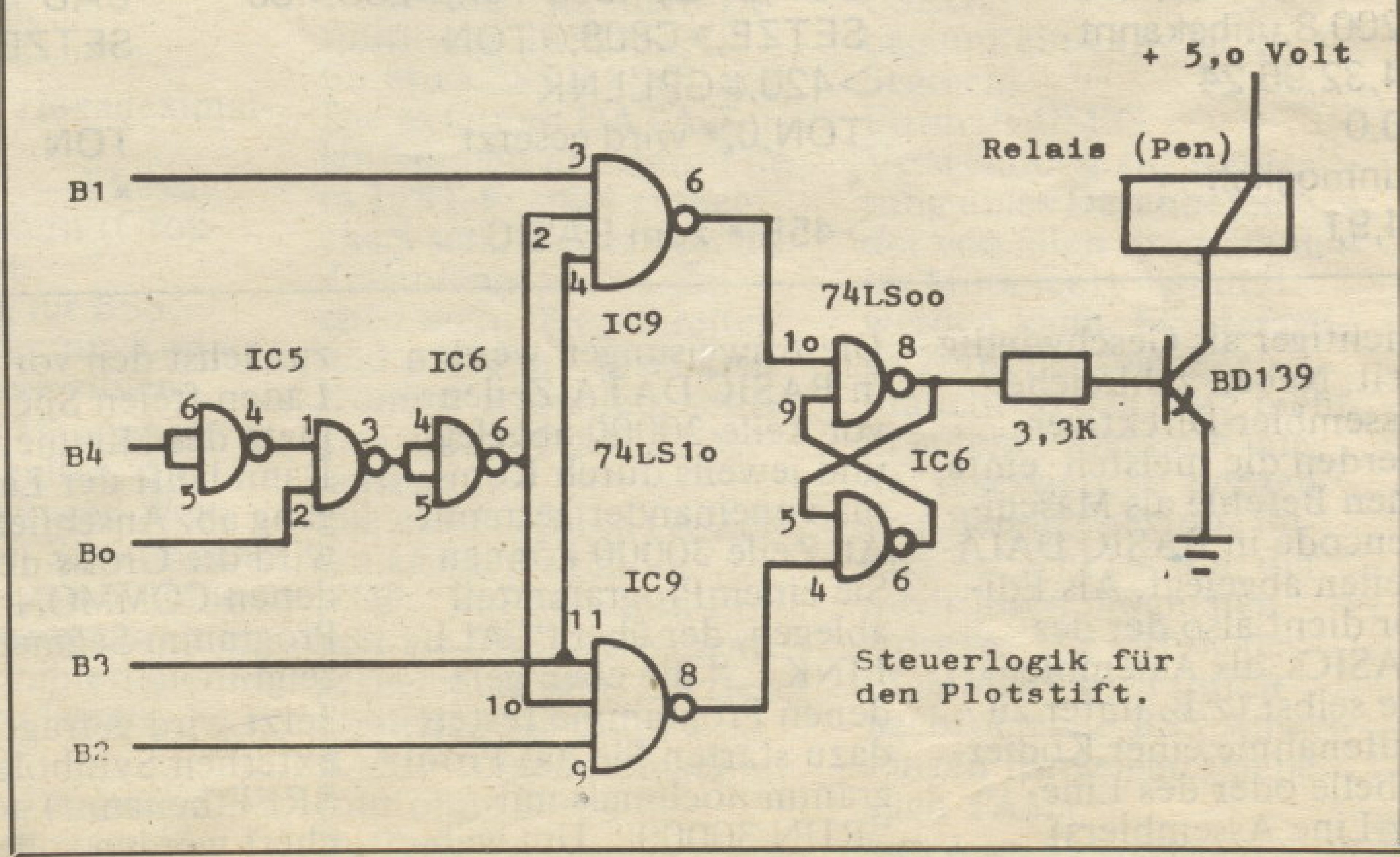
Wie aus der Tabelle zu ersehen ist, gibt es zwei Befehle für den Zeichenstift: einen zum Heben und einen zum Senken. Die beiden Befehle werden von IC5, IC6 und IC10 entschlüsselt und auf das mit zwei Gattern von IC6 gebildete Flip-Flop geschickt, dessen Ausgang das Relais über die Basis des BD139 steuert.

DER AUFBAU

Die Schaltung ist problemlos auf einer Euro-Lochrasterplatine aufzubauen. Alle IC's werden mit je einem 100nF Keramik Kondensator geschützt. Die Verbindungen zum Interface, zur Brücke und zu den Motoren habe ich mit mehradrigem Flachkabel hergestellt. Die Lichtschranken werden mit zweiadrigem abgeschirmtem Kabel mit der Schaltung verbunden.
Die Spannungsversorgung läßt sich, falls keine andere vorhanden ist, mit einem 7805 5 V-Spannungsregler mit Kühlblech leicht realisieren.
Im letzten Teil, im nächsten Heft, folgt dann die notwendige Software.

Mathias Wahner





CALL LOAD'S

TIPS & TRICKS

Die wichtigsten, bekannten und neuen, Poke's für den TI-99/4A mit EX.BASIC+32K Erweiterung:
Also aufgepaßt!

CALL LOAD(–31931,16)
= 'Trace' Befehl
CALL LOAD(–31931,2)
='On Warning Next'-Befehl
CALL LOAD(–31744,...)
= erzeugt einen Ton
CALL LOAD(–31878,50)
= löscht teilweise Buchstaben

CALL LOAD(–31878,1)
= macht dies rückgängig
CALL LOAD(–31878,0)
= stoppt alle Sprites
CALL LOAD(–31806,32)
= läßt einen 'Sound' ewig klingen
CALL LOAD(–31804,0,36)
= progr.mäßiges Aussteigen aufs Titelbild
CALL LOAD(–31932,0)
= 'Stop' Befehl ohne Ready-Meldung

CALL LOAD(–31806,16)
= schaltet die 'Quit'-Funktion aus
CALL LOAD(–31965,19)
= schaltet die Tastatur aus
CALL LOAD(–31806,0)
= schaltet die 'Quit'-Funktion ein
CALL LOAD(–31965,20)
= führt ein 'New' nach dem Editieren aus
CALL LOAD(–31965,21)
= 'New'-Befehl

CALL LOAD(–31931,128)
= 'Listschutz' ein
CALL LOAD(–31931,0)
= 'Listschutz' aus
CALL LOAD(–31931,200)
= 'On Break Next'-Befehl
CALL LOAD(–31903,210)
= Stoppt alles
CALL LOAD(–31806,33)
= nun ist kein Ton mehr da.

Anm. d. Red.: Da uns immer wieder Fragen erreichen: Der TI 99/4A wurde mit verschiedenen Betriebssystemen ausgeliefert, so müssen nicht alle hier aufgeführten Poke's die beschriebene Wirkung haben.

BESSER PROGRAMMIEREN MIT DEM M-LADER

ANLEITUNG ZUM M-LADER

Dieses Programm unterstützt die Programmierung in Maschinensprache und wurde speziell für die Konfiguration Minimemory mit Cassettenrecorder entwickelt. Dabei war Vielseitigkeit

Es stellt einen Kompromiß dar zwischen Assemblermodul. Line-by-Line Assembler und einfachen POKE-Listen. Den Unterschied verdeutlicht unser Beispiel, das 2 Programme lädt, die jeweils den "Accept-Ton" bzw. den "Bad-Response-Ton" erzeugen.

POKE-Liste (DATA Zeilen)	M-Lader Liste (DATA Zeilen)	TI Assembler		
unmöglich	GPLLNK EQ >6018	GPLLNK	EQU >6018	
unmöglich	* siehe weiter unten		DEF GOOD,BAD	
unmöglich	*	*		
unmöglich,2,8,0,52	DEF,GOOD,*Eingabeton,>208,>34	GOOD	LI R8,>34	Eingabeton
16,2	JMP SETZE		JMP SETZE	
unmöglich,2,8,0,54	DEF,BAD,* Bad-Ton,>208,>36	BAD	LI R8,>36	Bad-Ton
200,8,unbekannt	SETZE,>C808,@TON	SETZE	MOV R8,@TON	
4,32,96,24	>420,@GPLLNK		BLWP @GPLLNK	
0,0	TON,0,* wird gesetzt	TON	DATA 0	wird gesetzt
unmöglich	*	*		
4,91	>45B,* zum BASIC		B *R11	zum BASIC

wichtiger als Geschwindigkeit. Neben zahlreichen Assembler-Direktiven werden die meisten einfachen Befehle als Maschinencode in BASIC DATA-Zeilen abgelegt. Als Editor dient also der des BASICs, als Assembler Sie selbst (z.B. unter zu Hilfenahme einer Kodiertabelle oder des Line-by-Line Assemblers).

Die Anweisungen werden in BASIC DATA-Zeilen vor Zeile 20000 abgelegt und jeweils durch Komma voneinander getrennt. Ab Zeile 30000 können Sie einen Programmteil ablegen, der über "CALL LINK ..." die eben geladenen Programme testet; dazu starten Sie das Programm nochmals mit "RUN 30000". Um während des Ladevorganges alle zu verarbeitenden Anweisungen auf dem Bildschirm zu listen, muß die REM-Zeile 25500 aktiviert werden. Sie zeigt dann zusätzlich noch die FILE- und Zeilen-Nummer.

Das Programm wird mit RUN gestartet und zeigt zunächst den vor dem Laden freien Speicherplatz des Minimemory an. Dann läuft der Ladevorgang ab. Anschließend wird die Größe des geladenen COMMON- und Programm-Segmentes gezeigt.

Jetzt wird gefragt, ob die externen Symbole (über SREF benannt) gespeichert werden sollen, damit sie später beim Laden weiterer Programme noch aufgelöst oder benutzt werden können. Diese Möglichkeit gibt es bei Texas Instrument sonst nicht, erweist sich aber in der Minimalkonfiguration als äußerst nützlich. Nur falls Ihre Antwort mit einem der Buchstaben "N,n" beginnt, werden die Symbole nicht gespeichert.

Darauf werden zur Kontrolle alle Symbole gelistet, und zwar mit Name, Wert (meist Adresse), Bekanntheitsstatus, interner oder externer Typ und der Angabe, wie oft sie benutzt wurden.

Schließlich wird der nicht mehr ausgenutzte Speicher-

platz des Minimem angezeigt. Das Programm läuft danach in eine endlose GOSUB-Schleife, die mit "* Memory full*" endet. "PRINT FREI" zeigt dann den noch freien BASIC-Speicher an, d.h. um wieviel Ihre M-Lader Liste noch maximal wachsen darf.
Im Ablauf des Programms wird auf folgende Fehler geprüft:
dump. symbol
Änderung des bisherigen Wertes
no such symbol
Symbolname bei SREF noch unbekannt
CSEG into PSEG
COMMON größer als bisher gespeicherte Programme
memory overflow
Speicherplatz des Minimemory reicht nicht aus
JUMP reach error
bedingter Sprung über mehr als 256 Bytes
HEX code error
unerlaubtes Zeichen in einer Hexadezimalzahl
numeric overflow
Zahl zu groß für 1 Speicherwort bzw. -byte

Nach Ausgabe der Fehlermeldung kann der Ladevorgang zwar mit [ENTER] fortgesetzt werden, jedoch kann dies unvorhersehbare Folgen haben.

Welche Anweisungen und Direktiven stehen zur Verfügung?

Folgende Bezeichnungen werden benutzt:

ø	eine Leertaste
[ ]	muß durch die bezeichnete Größe ersetzt werden
LAH	Line-by-Line Assembler Handbuch des Minimemory
TAM	TI Editor/Assembler Manual

[xyz]
definiert ein Symbol mit Namen "xyz"
Format: [Text mit ASCII Wert> 64 am Anfang]
Alle Anweisungen außer den weiter unten aufgezählten werden als Definition eines Symbols aufgefaßt. Ihm wird die momentane Speicheradresse als Wert zugewiesen. Die maximal 40 Symbole dürfen beliebig lang sein, jedoch werden bei DEF und SREF nur die ersten 6 benutzt. Groß- und Kleinschreibung wird unterschieden.

@
benutzt als Wert den eines Symbols
Format: @[Symbol]
Der Wert des Symbols wird in 2 Bytes gespeichert. Das Symbol kann auch erst später definiert werden.

*
beginnt einen Kommentar
Format: *[Text in BASIC DATA-Zeilen Format]
Verwendung wie im TAM beschrieben, um die Eingabe mit Kommentaren zu strukturieren. Falls der Text Separatoren wie "," enthält, muß er einschließlich des Zeichens "*" in Anführung gesetzt werden. Außerdem können als Kommentar auch BASIC REM-Zeilen benutzt werden.

>
beginnt eine Hexadezimalzahl
Format: >[1 – 4 hexadezimale Ziffern (Großbuchstaben)]
Verwendung für BSS, Adressen oder zur Kodierung von Assembler-Befehlen als Maschinencode mit Tabellen, die auf hexadezimalen Werten aufbauen. Sie können ja auch den Line-by-Line Assembler benutzen, um sich die Assemblerbefehle übersetzen zu lassen.

[Zahl]
beginnt eine Dezimalzahl
Format: [Dezimalzahl; kann auch negativ sein]
Verwendung wie oben unter ">" beschrieben.

BSS
Block startet beim Symbol
Format: BSS,[Hexa-/Dezimalzahl, größer 0]
Verwendung wie im LAH beschrieben, jedoch ohne auf eine gerade Adresse zu prüfen.

BYTE
nächster Wert ist als ein Byte zu speichern
Format: BYTE,[(Hexa-)Dezimalzahl]
Verwendung, um die nächste Zahl als nur 1 Byte zu speichern. Ansonsten werden alle Zahlen als Worte (=2 Bytes) gespeichert.

CEND
beendet das COMMON Segment
Format: CSEG
Verwendung siehe CSEG

COPY
fügt Anweisungen ein, die außerhalb des M-Laders stehen
Format: COPY,BS,[dezimale VDP Segment Nummer, >0]
bzw.: COPY,[Filename (DATA-Zeilen in LIST-Format)]
Die Anweisung kann 9-fach geschachtelt sein. Die momentane Ebene der Schachtelung wird in Zeile 26600 sowie bei Fehlermeldungen als FILE-Nummer gezeigt.
Verwendung zum Einlesen aus externen Programmteilen, z.B. Standard-Definitionen über "EQU" für etwa "STRASG" u.ä.
Die externen DATA-Zeilen können in einem File in LIST-Format gespeichert sein. Dabei darf die Zeilenlänge ca. 65 Zeichen nicht überschreiten und die letzte Zeile kein Komma enthalten, sollte also z.B. "30000 REM" lauten.
Die externen DATA-Zeilen können auch im VDP in Programm-Format als BASIC-Segment gespeichert sein. Letzteres erreichen Sie unter BASIC folgendermaßen (nur ohne Disc-File-Buffer):

- Sie benutzen ein Standardverfahren, um ein BASIC-Programm im Speicher zu erhalten, das in DATA-Zeilen die M-Lader-Anweisungen enthält;
- Sie geben folgende Befehle ein

```
CALL PEEK
(-31888,A,S)
CALL PEEK
(-31952,Q,W,E,R)
ENDE=256*Q+W-7
END1=INT(ENDE/256)
END2=ENDE-256*
      END1
CALL LOAD
(-31888,END1,END2)
CALL POKEV
(ENDE+1,A,S,Q,W,E,R)
NEW
```

VDP-Speicher
BS.2: DATA Zeilen
BS.1: DATA Zeilen
BS.0: M-Lader mit COPY-Befehl

- Damit ist ein BASIC-Programm gespeichert und der jetzt reduzierte Speicher kann ganz normal das nächste Programm aufnehmen, z.B. den M-Lader durch "OLD CS1".

Jedes so gespeicherte Programm belegt ein BASIC-Segment (BS), durchnumeriert vom neuesten zum ältesten. Beachten Sie, daß bei Verwendung von Disketten-Laufwerken, deren I/O-Buffer stets als erstes Segment existieren muß, d.h. als "BS.1".

CSEG
beginnt ein COMMON Segment
Format: CSEG
Verwendung zur Reservierung eines Datenbereiches, der von allen Programmen im Minimeory benutzt werden kann. Für derartige Zwischenspeicher braucht so der Platz nicht in jedem Programm wieder reserviert zu werden. Der Bereich beginnt immer bei >7118, d.h. der niedrigsten möglichen Adresse. Die Größe des COMMON wird durch nachfolgende BSS Anweisungen festgelegt; (siehe auch TAM).

Bitte lesen Sie weiter auf S. 16

LISTINGS

```
300 REM BEISPIEL FUER
310 REM M/LADER :
320 REM
330 REM Routinen GOOD
340 REM      und BAD
350 REM
360 REM ------------------

370 REM
380 DATA GPLLNK EQU >6018
390 DATA * siehe weiter unte
n
400 DATA *
410 DATA DEF,GOOD,* Eingabet
on,>208,>34
420 DATA JMP SETZE
430 DATA DEF,BAD,* Bad-Ton,>
208,>36
440 DATA SETZE,>C808,@TON
450 DATA >420,@GPLLNK
460 DATA TON,0,* wird gesetz
t
470 DATA *
480 DATA >45B,* zum BASIC
490 REM
500 REM -------------------
20000 REM M-Lader        Md
20005 REM stat. 25.jan.1986
          back. 25.jan.1986
20010 REM (C) G. Zellermann
           1986    Koeln
20020 DATA END,* fuer verges
sliche hacker
20030 RESTORE 20030
20040 OPTION BASE 1
20050 DIM B(6),DIR_NAME$(13)
20060 DIM S_NAME$(40),S_ADR(
40),S_STATUS$(2,40),S_REF(40
)
20070 DATA BYTE,DEF,SREF,LSE
G,END,BSS,EVEN,TEXT,TEXT_B,C
SEG,CEND,COPY,RETURN,""
20080 READ C$
20090 IF C$="" THEN 20130
20100 DIR_MAX=DIR_MAX+1
20110 DIR_NAME$(DIR_MAX)=C$
20120 GOTO 20080
20130 DEF I5_BY$(X)=SEG$("
   ",1,5-LEN(STR$(X)))&STR$(X
)&" bytes"
20140 DEF BL(C)=INT(C/256)
20150 DEF BR(C)=C-256*BL(C)
20160 CALL PEEK(28700,AL,AR,
CL,CR)
20170 FFAM=256*AL+AR
20180 LFAM=256*CL+CR
20190 PRINT : :"   vorher  f
rei :";I5_BY$(LFAM-FFAM):""
20200 REM - get old symb. -
20210 FOR ADR=LFAM TO 32767
STEP 8
20220 CALL PEEK(ADR,B(1),B(2
),B(3),B(4),B(5),B(6),CL,CR)
20230 IF B(2)<128 THEN 20410
20240 S_MAX=S_MAX+1
20250 LFAM=LFAM+8
20260 CALL LOAD(28702,BL(LFA
M),BR(LFAM))
20270 B(2)=B(2)-128
20280 S_STATUS$(2,S_MAX)="ex
t"
20290 S_STATUS$(1,S_MAX)="ja
"
20300 IF B(1)<128 THEN 20330
20310 B(1)=B(1)-128
20320 S_STATUS$(1,S_MAX)="NO
"
20330 S_ADR(S_MAX)=256*CL+CR
20340 S_REF(S_MAX)=0
20350 C$=""
20360 FOR CH=1 TO 6
20370 IF B(CH)=32 THEN 20400
20380 C$=C$&CHR$(B(CH))
20390 NEXT CH
20400 S_NAME$(S_MAX)=C$
20410 NEXT ADR
20420 ADR=FFAM
20430 PSEG_ANF=ADR
20440 CSEG_ADR=28952
20450 IF LSEG THEN 25950
20460 GOSUB 26000
21000 REM -- lese codes --
21010 GOSUB 26400
21020 REM -- type of code -
21030 IF ASC(C$)<>62 THEN 21
060
21040 GOSUB 22700
21050 GOTO 21000
21060 IF ASC(C$)=42 THEN 210
00
21070 FOR DIR=1 TO DIR_MAX
21080 IF C$=DIR_NAME$(DIR)TH
EN 21100
21090 NEXT DIR
21100 ON DIR GOSUB 22500,242
00,24500,25300,25500,24000,2
4100,24800,24800,25000,2
5100,26700,26100,21200
21110 GOTO 21000
21200 REM -- no direct. --
21210 ASSEM=POS(C$," ",1)
21220 IF ASSEM=0 THEN 21280
21230 IF "EQU "=SEG$(C$,ASSE
M+1,4)THEN 21500
21240 ASSEM$=SEG$(C$,2,ASSEM
-1)
21250 C$="@"&SEG$(C$,ASSEM+1
,251)
21260 ASSEM=POS("  MP LT LE
```

```
EQ HE GT NE NC OC NO L  H  O
P",ASSEM$,3)
21270 ASSEM=15+ASSEM/3
21280 REM old part / no J...
21290 KNOW=2+SGN(64-ASC(C$))
21300 ON KNOW GOTO 23220,232
00,22700
21500 REM -- EQU symbol ---
21510 LABEL$=SEG$(C$,1,ASSEM
-1)
21520 C$=SEG$(C$,ASSEM+5,250
)
21530 GOSUB 22000
21540 PSEG_ADR=ADR
21550 ADR=C
21560 C$=LABEL$
21570 GOSUB 21280
21580 ADR=PSEG_ADR
21590 RETURN
22000 REM - wandle numerisch
22010 IF ASC(C$)<62 THEN 221
40
22020 REM -- hexa. zahl --
22030 C=0
22040 FOR CH=2 TO LEN(C$)
22050 HEX=ASC(SEG$(C$,CH,1))
-48
22060 IF HEX<10 THEN 22080
22070 HEX=HEX-7
22080 IF (-1<HEX)*(HEX<16)TH
EN 22110
22090 PRINT :"** HEX-code er
ror";
22100 GOSUB 29900
22110 C=C*16+HEX
22120 NEXT CH
22130 GOTO 22170
22140 C=VAL(C$)
22150 IF C>=0 THEN 22170
22160 C=C+65536
22170 CL=INT(C/256)
22180 CR=C-256*CL
22190 RETURN
22500 REM -- BYTE option --
22510 INC=1
22520 GOSUB 26400
22530 GOTO 22720
22700 REM - speichere code -
22710 INC=2
22720 GOSUB 22000
22730 GOSUB 29000
22740 GOSUB 29300
22750 CALL LOAD(ADR-1,CR)
22760 IF INC=1 THEN 22780
22770 CALL LOAD(ADR-2,CL)
22780 RETURN
23000 REM -- find symbol --
23010 FOR S=1 TO S_MAX
23020 IF S_NAME$(S)=C$ THEN
23060
```

LISTINGS

```
23030 NEXT S
23040 S_STATUS$(1,S)=""
23050 S_REF(S)=0
23060 RETURN
23200 REM --- SYMBOL ---
23210 C$=SEG$(C$,2,254)
23220 REM - entry for def. -
23230 GOSUB 23000
23240 KNOW=KNOW-4*(S>S_MAX)-
2*(S_STATUS$(1,S)="NO")
23250 ON KNOW GOTO 23380,233
40,23340,23340,23290,23260
23260 REM -- new SYMBOL --
23270 CR=(ASSEM>15)
23280 CL=-ASSEM*CR
23290 S_NAME$(S)=C$
23300 S_STATUS$(2,S)="int"
23310 S_STATUS$(1,S)=SEG$("j
aNO",2*KNOW-9,2)
23320 S_MAX=S
23330 GOTO 23410
23340 REM -- old SYMBOL --
23350 CL=BL(S_ADR(S))
23360 CR=BR(S_ADR(S))
23370 IF KNOW=2 THEN 23420 E
LSE 23410
23380 REM -- warnung --
23390 PRINT :"** dupl.";
23400 GOSUB 29200
23410 S_ADR(S)=ADR
23420 ON KNOW GOTO 23580,235
90,23430,23590,23580,23630
23430 REM - alte aufloes. -
23440 S_STATUS$(1,S)="ja"
23450 GOTO 23490
23460 CALL PEEK(C,CL,CR)
23470 IF (15<CL)*(CL<29)THEN
 23510
23480 CALL LOAD(C,BL(ADR),BR
(ADR))
23490 C=256*CL+CR
23500 IF C THEN 23460 ELSE 2
3580
23510 REM relative adr./Jxx
23520 CD=CR-255
23530 CR=(ADR-C-2)/2
23540 GOSUB 29100
23550 CALL LOAD(C+1,CR)
23560 C=C+2*CD
23570 IF CD THEN 23460
23580 RETURN
23590 IF ASSEM<16 THEN 23630
23600 CR=(256*CL+CR-ADR-2)/2
23610 CL=ASSEM
23620 GOSUB 29100
23630 S_REF(S)=S_REF(S)+1
23640 INC=2
23650 GOTO 22730
24000 REM -- BLOCK frei --
24010 GOSUB 26400
```

```
24020 GOSUB 22000
24030 INC=C
24040 GOTO 29000
24100 REM -- gerade adr. --
24110 ADR=2*INT((ADR+1)/2)
24120 RETURN
24200 REM -- DEF eintr. --
24210 KNOW=1
24220 GOSUB 26400
24230 GOSUB 23220
24240 NAME$=C$
24250 INC=-8
24260 GOSUB 29000
24270 REM - entry from SREF
          postprocessing
24280 C$=SEG$(NAME$&"      ",
1,6)
24290 T=LFAM
24300 BIAS=0
24310 CALL LOAD(28702,BL(LFA
M),BR(LFAM))
24320 CALL LOAD(LFAM+6,BL(AD
R),BR(ADR))
24330 GOTO 24700
24500 REM -- SREF kennz. --
24510 GOSUB 26400
24520 GOSUB 23000
24530 IF S<=S_MAX THEN 24560
24540 PRINT :"** no such";
24550 GOSUB 29200
24560 S_STATUS$(2,S)="ext"
24570 RETURN
24700 REM -- lade ASCII --
24710 FOR CH=1 TO LEN(C$)
24720 C=BIAS+ASC(SEG$(C$,CH,
1))
24730 CALL LOAD(T-1+CH,C)
24740 NEXT CH
24750 RETURN
24800 REM -- TEXT eintr. --
24810 BIAS=-96*(C$="TEXT_B")
24820 T=ADR
24830 GOSUB 26400
24840 INC=LEN(C$)
24850 GOSUB 29000
24860 GOTO 24700
25000 REM -- CSEG direk. --
25010 PSEG_ADR=ADR
25020 ADR=CSEG_ADR
25030 RETURN
25100 REM -- CEND direk. --
25110 CSEG_ADR=ADR
25120 IF PSEG_ANF>=ADR THEN
25180
25130 IF PSEG_ANF<PSEG_ADR T
HEN 25160
25140 PSEG_ANF=ADR
25150 GOTO 25190
25160 PRINT "** CSEG into PS
EG";
25170 GOSUB 29900
25180 ADR=PSEG_ADR
25190 RETURN
25300 REM ---  LSEG  ---
25310 FORGET_IT=0
25320 GOSUB 25950
25500 REM -- ende  PSEG --
25510 FFAM=ADR
25520 INC=-8
25530 CALL LOAD(28700,BL(ADR
),BR(ADR))
25540 PRINT : :"COMMON-segme
nt     ";I5_BY$(CSEG_ADR-28952
)
25550 PRINT "PROGRAMM-segmen
t ";I5_BY$(FFAM-PSEG_ANF):""
25560 IF LSEG THEN 25590
25570 INPUT "Externe Symbole
 speichern ":C$
25580 FORGET_IT=-POS("Nn",SE
G$(C$&"Y",1,1),1)
25590 REM - liste symbole -
25600 PRINT : :" Name  Adres
se bekannt #Ref.------------
----------------"
25610 FOR S=1 TO S_MAX
25620 PRINT S_NAME$(S);
25630 PRINT TAB(8);S_ADR(S);
25640 PRINT TAB(16);S_STATUS
$(1,S);"/";S_STATUS$(2,S);
25650 PRINT TAB(24);S_REF(S)
25660 IF FORGET_IT+(S_STATUS
$(2,S)="int")THEN 25740
25670 ADR=FFAM
25680 GOSUB 29000
25690 ADR=S_ADR(S)
25700 NAME$=S_NAME$(S)
25710 GOSUB 24270
25720 CALL PEEK(LFAM,CL,CR)
25730 CALL LOAD(LFAM,CL-128*
("NO"=S_STATUS$(1,S)),CR+128
)
25740 NEXT S
25750 PRINT :"    nachher fre
i :";I5_BY$(LFAM-FFAM): :""
25760 IF LSEG THEN 25900
25800 FREI=FREI+8
25810 GOSUB 25800
25900 REM --- prep. for
              next module
25910 S_MAX=0
25920 GOTO 20200
25950 REM -- switch LSEG
25960 LSEG=1-LSEG
25970 RETURN
26000 REM  ==intern. DATA==
26010 CALL PEEK(-31952,A1,A2
,E1,E2)
26020 COPY=1
26030 CALL PEEK(-31888,AL,AR
```

```
)
26040 C=256*AL+AR+1
26050 REM new V-BASIC file
26060 DATA_BYTE(COPY)=C-2
26070 L_TAB_END(COPY)=256*A1
+A2
26080 L_TAB(COPY)=256*E1+E2+
1
26090 RETURN
26100 REM - close V-BASIC -
26110 IF L_TAB_END(COPY)>0 T
HEN 26130
26120 CLOSE #COPY
26130 COPY=COPY-1
26140 IF COPY THEN 26610
26150 STOP
26200 REM next BASIC-line
26210 CALL PEEKV(L_TAB(COPY)
,A1,A2,E1,E2)
26220 DATA_BYTE(COPY)=256*E1
+E2
26230 GOSUB 26640
26240 IF E1=147 THEN 26310
26250 REM more BASIC-lines
26260 L_TAB(COPY)=L_TAB(COPY
)-4
26270 IF L_TAB(COPY)>=L_TAB_
END(COPY)THEN 26200
26280 C$="RETURN"
26290 DATA_LINE(COPY)=99999
26300 GOTO 26600
26310 REM new  DATA-line
26320 DATA_LINE(COPY)=256*A1
+A2
26400 REM -- next DATA-item
26410 IF L_TAB_END(COPY)<1 T
HEN 27000
26420 GOSUB 26620
26430 IF E1=0 THEN 26250
26440 IF E1=179 THEN 26400
26450 GOSUB 26620
26460 C$=""
26470 FOR E2=1 TO E1
26480 GOSUB 26620
26490 C$=C$&CHR$(E1)
26500 NEXT E2
26600 REM PRINT STR$(COPY);T
AB(7-LEN(STR$(DATA_LINE(COPY
))));DATA_LINE(COPY);" "
;C$
26610 RETURN
26620 REM read 1 byte
26630 DATA_BYTE(COPY)=DATA_B
YTE(COPY)+1
26640 CALL PEEKV(DATA_BYTE(C
OPY),E1)
26650 RETURN
26700 REM -- next COPY-level
26710 GOSUB 26400
26720 COPY=COPY+1
```

LISTINGS

```
26730 IF POS(C$,"BS.",1)=1 T
HEN 26850
26740 OPEN #COPY:C$,DISPLAY
,INPUT
26750 INPUT #COPY:C$
26760 GOSUB 26030
26770 L_TAB(COPY)=0
26780 L_TAB_END(COPY)=0
26790 DATA_BYTE(COPY)=C+9
26800 FOR C=2 TO COPY-1
26810 IF DATA_BYTE(COPY)<>DA
TA_BYTE(C)THEN 26830
26820 DATA_BYTE(COPY)=DATA_B
YTE(COPY)+518
26830 NEXT C
26840 RETURN
26850 VSEG=VAL(SEG$(C$,4,LEN
(C$)))
26860 CALL PEEK(-31888,AL,AR
)
26870 FOR I=1 TO VSEG
26880 C=256*AL+AR+1
26890 IF C>16383 THEN 26100
26900 CALL PEEKV(C,AL,AR,A1,
A2,E1,E2)
26910 IF AL-170 THEN 26940
26920 C=C+1
26930 GOTO 26890
26940 NEXT I
26950 GOTO 26040
26960 REM mark non-DATA
26970 L_TAB(COPY)=-1
27000 REM item from disk
27010 IF EOF(COPY)THEN 26280
27020 INPUT #COPY:C$,
27030 GOSUB 26640
27040 IF E1+L_TAB_END(COPY)=
0 THEN 27120
27050 L_TAB_END(COPY)=-E1
27060 IF E1=0 THEN 27120
27070 C=POS(C$," DATA ",2)
27080 IF (C=0)+(C>6)THEN 269
60
27090 L_TAB(COPY)=0
27100 DATA_LINE(COPY)=VAL(SE
G$(C$,1,C-1))
27110 C$=SEG$(C$,C+6,75)
27120 IF L_TAB(COPY)THEN 270
00 ELSE 26600
29000 REM -- check room --
29010 IF LFAM-ADR>ABS(INC)TH
EN 29040
29020 PRINT :"** memory over
flow";
29030 GOSUB 29900
29040 IF INC>0 THEN 29070
29050 LFAM=LFAM+INC
29060 RETURN
29070 ADR=ADR+INC
29080 RETURN
```

```
29100 REM -- check jump --
29110 IF (-128<CR)*(CR<129)T
HEN 29080
29120 PRINT :"** JUMP reach
error";
29130 GOTO 29900
29200 REM warnung wg.SYMBOL
29210 PRINT " symbol ";
29220 GOTO 29900
29300 REM check byte-bound.
29310 IF CL<256^(INC-1)THEN
29080
29320 PRINT :"** numeric ove
rflow";
29900 REM err. data line-#
29910 PRINT TAB(20);"**":"
 kode ist : ";C$:"   file/ze
ile ";STR$(COPY);"/";STR
$(DATA_LINE(COPY))
29920 INPUT " ":T$
29930 RETURN
30000 REM ---  test it  ---
```

Fortsetzung von S. 11

Da nach CALL INIT ab hier auch die Assemblerprogramme geladen werden, muß der Abschnitt CSEG bis CEND vor dem eigentlichen Programm stehen. Zudem muß das erste geladene Programm bereits eine, später benötigte, maximale Größe für den COMMON reservieren.

(DATA)
wird nicht benutzt!!
Alle (Hexa-)Dezimalzahlen werden ja sowieso direkt als Worte gespeichert. Dies darf nicht mit der BASIC DATA Anweisung verwechselt werden, die ja schon benutzt wird, um alle M-Lader Anweisungen zu halten.

DEF
definiert Symbol als Programmnamen, auch in REF/DEF Tabelle
Format: DEF,[Symbol]
Verwendung zur Zuweisung der momentanen Ladeadresse als Wert und Eintragung des Symbols in die REF/DEF Tabelle. Sie unterscheidet sich also von der im TAM beschriebenen Direktive, weil jene am Anfang des Programms stehen muß und nicht wie hier an der Stelle, die tatsächlich als Programmstartadresse gemeint ist.

END
beendet den Ladevorgang
Format: END (nicht unbedingt nötig)
Verwendung als Abschluß des Ladevorgangs. Da der M-Lader am Anfang in Zeile 20020 sicherheitshalber selbst eine DATA-Zeile mit "END" enthält, ist diese Anweisung nicht nötig, kann aber bei der Fehlersuche hilfreich sein, um nur einen Teil des Assembler-Programms zu laden.

EVEN
geht zur nächsten geraden Adresse vor
Format: EVEN
Verwendung wie im TAM beschrieben, z.B. nach BSS, BYTE, TEXT-B

EQU
setzt ein Symbol auf einen Wert
Format: [Symbol]ØEQUØ[Hexy- oder Dezimalzahl]
Verwendung wie im LAH beschrieben, jedoch nur, um Zahlenwerte zuzuordnen. Nur hier wird also dem Symbol nicht die momentane Ladeadresse zugeordnet.

Jx(y)
bedingter Sprung zu einem Label
Format: [Assembler JUMP Anweisung]Ø[Symbol]
Verwendung für jeden der 13 Sprungbefehle möglich, jedoch kann das Ziel nur als Symbol angegeben werden.

LSEG
wirkt wie ein neuer Start des M-Laders
Format: LSEG
Verwendung um mehrere Programme direkt nacheinander zu laden. Werden sie durch "LSEG" getrennt, so ist sichergestellt, daß die einfachen Symbole für jedes Programm separat behandelt werden. Für gemeinsame Symbole wird "SREF" benutzt.
Diese Anweisung ist besonders für Disketten gedacht, indem ein File die Namen aller zu ladenden Programme enthält, wobei jene jeweils über "COPY" aufgerufen und durch "LSEG" voneinander getrennt werden.

RETURN
beendet die Wirkung von COPY
Format: RETURN
Verwendung kann entfallen am Ende des jeweiligen Speicherbereichs; denn am Ende wird automatisch ein RETURN ausgeführt, das dann im Listing unter Zeilen-Nummer "99999" erscheint.

SREF
bezeichnet ein Symbol als extern; Eintrag in REF/DEF Tabelle
Format: SREF,[Symbol] (Symbol muß schon benutzt sein)
Verwendung bei Symbol, das in einem anderen Assembler-Programm bei einem separaten M-Lader Lauf definiert (d.h. aufgelöst) oder benutzt wird, z.B. als Adresse einer gemeinsamen Routine.
Somit können mit dem M-Lader auch größere Programm-Bibliotheken zusammengestellt werden.
Siehe auch TAM.

Solange ein solches externes Symbol noch unbekannt ist, sind alle geladenen Assemblerprogramme, auch die alten, blokkiert.

TEXT
lädt Text als ASCII-Folge für Assembler-Programme
Format: TEXT,[Text-String in BASIC DATA-Zeilen Format]
Verwendung wie im LAH beschrieben, jedoch ohne auf eine gerade Adresse zu prüfen. Verwendung nicht für Programme, die als Routinen im BASIC verwendet werden, und zwar wegen des Screen-Offset.

TEXT_B
lädt Text als ASCII-Folge für BASIC-Mode
Format: TEXT_B, [Text-String in BASIC DATA-Zeilen Format]
Verwendung wie im LAH beschrieben, jedoch ohne auf eine gerade Adresse zu prüfen. Verwendung nur für Programme, die als Routinen im BASIC verwendet werden, und zwar wegen des Screen-Offset.

Dr. G. Zellermann

ANMERKUNG ZU M-LADER

M-Lader ist ein Hilfsprogramm, das hauptsächlich für den Anwender mit Mini-Mem und Kassettenrecorder geschrieben wurde. Die einzige Routine, die auch an den Diskettenbetrieb angepaßt ist, ist die 'COPY' Anweisung. Es ist jedoch nur möglich, das Programm mit einem Corcomp-Controller für Diskettenlaufwerk zu betreiben, da ansonsten trotz 'CALL FILES(1)' der Speicher zum Betrieb dieses Programmes nicht ausreicht. Beim Abtippen ist ebenfalls darauf zu achten, daß alle Underlines ('_'), die teilweis bei Variablennamen verwendet wurden, richtig gesetzt werden.

Die Redaktion

SO FRAGEN SIE IHRE TASTATUR AB



Sehr oft schon wurde gefragt, ob man die Tastatur mittels Call peek abfragen kann. Daß dies nicht geht, da die Tastatur von der CRU abgefragt wird, dürften mittlerweile alle TI-User wissen, die sich mit diesem Problem beschäftigt haben. Beim Suchen von nützlichen Adressen stieß ich nun auf die Adressen –31798, –31799 und –31800. Diese Adressen lösen das Problem nicht vollständig, doch helfen sie, die Tastaturabfrage zu verfeinern. Ein Call PEEK (–31800,X) weist jeder gedrückten Taste in X einen Wert zu (dieser Wert stimmt nicht mit den ASCII-Codes überein). Dabei ist es egal, ob Sie eine Taste in Verbindung mit FCTN, CTRL oder SHIFT gedrückt haben. Die Taste behält ihren Wert immer bei.

Call Peek (–31800,X) fragt die gesamte Tastatur ab. Wenn Sie nun vor dem Call Peek ein Call Key durchführen, können Sie ständig jede Taste abfragen.

Call Peek (–31798,X) fragt nur die rechte Seite der Tastatur ab. Wird eine Taste der linken Seite gedrückt, wird dieser Taste der Wert 255 zugewiesen.

Call Peek (–31799,X) fragt die linke Tastaturseite ab. Hier wird der rechten Seite der Wert 255 zugewiesen. Die Werte, die den Tasten bei der Adresse –31798 und –31799 zugewiesen werden, stimmen mit den Werten der Adresse –31800 überein, außer, daß bei –31798 auch der Enter-, der Space- und der =-Taste der Wert 255 zugewiesen wird. Auch bei –31799 wird der Space-Taste der Wert 255 zugewiesen.

Diese Werte sind bei allen Tastaturmodi gleich, also bei Call Key (0–5,K,S). Eine Call Key-Abfrage, die die Adresse –31800 nützt, müßte etwa folgendermaßen aussehen:

```
100 Call Key (0,K,S)::
If S20 then 100 else Call
Peek (–31800,X)
110 weiteres Programm
```

Dirk Junghans

Bemerkung zu „Tastaturabfrage mit PEEK":

Auf den Adressen> 83C9 und> 83CA (–31800, –31799 und –31798) sind Kennwerte der Tastaturabfrage im ROM. Aus diesem Grund stimmen die Werte mit 'CALL PEEK' nicht mit den ASCII-Werten der betreffenden Taste überein.

Hier nun die Codes, die einer Taste bei einem Call Peek (–31800,X), Call Peek (–31798,X) oder Call Peek (–31799,X) zugewiesen werden.

	>83C8 –31800	>83C9 –31799	>83CA –31798
1=	43	43	255
2=	11	11	255
3=	19	19	255
4=	27	27	255
5=	35	35	255
6=	36	255	36
7=	28	255	28
8=	20	255	20
9=	12	255	12
0=	44	255	44
= =	7	255	255
A=	42	42	255
B=	32	32	255
C=	16	16	255
D=	18	18	255
E=	17	17	255
F=	26	26	255
G=	34	34	255
H=	38	255	38
I =	21	255	21
J=	30	255	30
K=	22	255	22
L=	14	255	14
M=	31	255	31
N=	39	255	39
O=	13	255	13
P=	45	255	45
Q=	41	41	255
R=	25	25	255
S=	10	10	255
T=	33	33	255
U=	29	255	29
V=	24	24	255
W=	9	9	255
X=	8	8	255
Y=	37	255	37
Z=	40	40	255
/ =	47	255	255
; =	46	255	46
, =	23	255	23
. =	15	255	15
ENTER=	5	255	255
Space =	6	255	255
Keine Taste gedrückt	255	255	255

WIZARD CASTLE

Als Sie aufwachen, befinden Sie sich in einem Ihnen unbekannten Raum. Langsam erinnern Sie sich. Sie hatten vor, die Tochter des Königs aus den Händen der gestaltlosen Handlanger des Zauberers zu befreien. Statt dessen befinden Sie sich nun hier, im Schloß des Zauberers, im Wizard Castle. Doch so leicht geben Sie nicht auf, für jedes Rätsel gibt es eine Lösung, nur, sie zu finden, ist jetzt das Problem. Sie ahnen, welche gefährlichen Situationen und Kämpfe auf Sie zukommen können. Man erzählt sich von blutrünstigen Bestien und Drachen, von räuberischen Wächtern und harmlos ausschauenden Kobolden, die hier im Schloß ihr Unwesen treiben sollen. Nichtsdestoweniger machen Sie sich auf den Weg, in der Hoffnung, Wizard Castle sein Geheimnis zu entreißen und lebend das Rätsel des Ausganges zu lüften.
Doch Vorsicht, geschwächt durch den vorangehenden Kampf sind Sie leider nur in der Lage, lediglich zwei Gegenstände mit sich zu tragen. Aber Sie als überlegener und entschlossener Kämpfer wissen um die richtige Auswahl der nötigen Objekte. Weiterhin sollten Sie sich immer im Klaren darüber sein, wo Sie sich zur Zeit befinden. Merken Sie sich Gegenstände, die Sie liegenlassen und prägen Sie sich vor allem die Lage der Auf- und Abstiege ein. Oder noch besser, Sie zeichnen sich einen Plan, ansonsten könnten Sie sich hoffnungslos verlaufen.
Und wer möchte dieses Risiko mit der Erinnerung an die Tochter des Königs schon eingehen?
Möge die Macht mit Ihnen sein! (Disk-Besitzer müssen vor dem Eintippen oder Laden CALL FILES (1) NEW eingeben!

ERLÄUTERUNGEN ZUM SPIELVERLAUF

Zu Beginn des Spieles befindet der Spieler sich im ersten Stock vorne rechts. In dieser Etage droht ihm meist noch kein Ärger. Die nötigen und tragbaren Gegenstände sind in den unteren drei, die gefährlichen Situationen in den oberen drei Etagen plaziert. Und so kommt man der Lösung auf die Spur: Finden Sie zunächst einmal den Schlüssel, der selbstverständlich schwer bewacht wird. Nehmen Sie ihn und legen diesen in einem Raum ab, da viele Räuber und geldgierige Wächter umherstrolchen und nur darauf warten, Sie auszurauben. Sodann suchen Sie den Raum mit der Truhe, öffnen sie mit dem vorher geholten Schlüssel – Eingabe: "öffne Truhe" – nachdem Sie die Wachen bezwungen haben. Sie sehen nun ein Buch, lassen Sie es am Besten dort und merken sich, welcher Raum dies ist. Jetzt sollten Sie den geheimnisvollen Schriftraum aufsuchen. Überwältigen Sie die Wache, holen Sie das Buch und erfahren somit den unachtsam hingekritzelten Zauberspruch – Eingabe: "übersetze" –. Diesen sechssilbigen Zauberspruch sollten Sie sich gut merken, nur mit ihm im Gedächtnis sind Sie befähigt, das magische Ausgangstor zu öffnen. Haben Sie schließlich den Ausgangsraum gefunden, sprechen Sie das magische Wort – Eingabe: "Zauberspruch" – und Sie haben die Qualen der letzten Nacht hinter sich. Auf zu neuen Taten!
Ergänzend sollen nachstehend noch einige Spielphasen erläutert werden.
An den Einzelwachen kommen Sie mittels des Beiles vorbei – Eingabe: "verjage Wache". Bei Doppelwachen ist dies zwecklos. Die Brüder sind aber enorm geldgierig, deshalb muß man ihnen das Gold überlassen – Eingabe: "gib Gold". In einigen Räumen erscheinen Kobolde. Vorsicht, nur der rechts stehende ist harmlos. Er läßt sich mitnehmen und dem Drachen oder dem Bär als Futter vorwerfen – Eingabe: "gib Kobold". Dem feuerspeienden Drachen kann man aber ebensogut mit dem Schild begegnen – Eingabe: "benutze Schild". Dementsprechend gibt sich der Bär auch mit dem Fleisch zufrieden – Eingabe: "gib Fleisch". Der linksstehende Kobold nun bewacht meist unsichtbare Türen. Mittels der Tarnkappe kann man ungehindert an ihm vorbeikommen – Eingabe: "benutze Tarnkappe". Allerdings sollte man dann wirklich westwärts gehen, da ansonsten die Wirkung der Tarnkappe erlischt. Manchmal wird dem Spieler der Weg auch durch eine Mauer versperrt. Diese bewältigt man einfachhalber mit einer Bombe – Eingabe: "zünde Bombe". Vorteilhaft ist es auch, sich ständig eine Fackel griffbereit zu halten, da es im Schloß ständig und oftmals dunkel wird. In dieser Dunkelheit lauern Räuber und Wächter, die Sie ausrauben wollen.
Da diese aber das Licht fürchten, sollten Sie für Helligkeit sorgen – Eingabe: "benutze Fackel". Sie können den Räubern aber auch Tribut zahlen, indem Sie ihnen ihr Gold überlassen – Eingabe: "gib Gold". Haben Sie diesen unliebsamen Weggenossen nichts entgegenzusetzen, werden Sie ausgeraubt und ins Schloß verschleppt. Mit diesen Hinweisen sollten Sie eigentlich in der Lage sein, das Abenteuer zumindest einmal zu durchqueren. Viel Glück und Verstand!

BEMERKUNGEN ZUR REALISATION

Zu Beginn der Programmentwicklung stellte ich – nicht überrascht – fest, daß der Speicherplatz des TI für diese umfangreiche Aufgabe einfach zu klein ist. Die von mir benutzte Lösung hatte nun eine Programmtrennung zur Folge. Im ersten Programm – Wizard Castle 1 – werden die Characters der Gegenstände definiert. Weiterhin ist hier die Spielanleitung enthalten. Dieses Pro-

gramm lädt selbständig – mittels RUN "CS1" – das Programm Wizard Castle 2. Solange nun dieses Programm läuft, bleiben die Zeichen erhalten. Nach Abbruch und nochmaligem Start kann man allerdings nicht mehr viel erkennen. In dieser Manier bleibt im Hauptprogramm genügend Platz für sämtliche Routinen usw.
Als Spielfeld für ein Abenteuerspiel empfahl sich mir ein dreidimensionales Array, in dem jedes Feld für einen Raum steht und in dem der Spieler sich von Raum zu Raum bewegen kann. Geht der Spieler zum Beispiel nach oben, so wird die Z-Koordinate um 1 erhöht, bewegt er sich nach unten, wird sie um 1 erniedrigt. Dieses System ist vor allem für die Simulation von Gebäuden günstig. In dieser Weise wird im vorliegenden Programm ein vierstöckiges Gebäude mit jeweils 4x4 Räumen in jeder Etage, insgesamt also 64 Räumen, simuliert. Der Inahtl der einzelnen Felder wird durch den Array-Inhalt gespeichert. Jedes Feld enthält eine Zahl, die, wenn sie bitweise betrachtet wird, Informationen über viele Objekte speichern kann. Dies geschieht in BASIC mit den logischen Operatoren AND und OR.
Das Ergebnis der Verknüpfung A AND B ist der Wert der Bits, die in beiden Zahlen A und B enthalten sind, zum Beispiel 12 AND 8 = 8.
Bitmuster der Ausgangszahlen 12: I100
8: I000.
In beiden Zahlen enthaltenes Bitmuster: I000=8.
Eine OR Verknüpfung hat als Ergebnis den Wert des Bits, die in einer oder in beiden Zahlen enthalten sind, zum Beispiel
2 OR 8 = 10. Bitmuster der Ausgangszahlen
2: 0010
8: I000.
In beiden Zahlen enthaltenes Bitmuster: I0I0=10.
Das vorliegende Programm sollte nun folgenden Anforderungen, genügen: Zum einen muß der Array-Inhalt die möglichen Türen, Auf- bzw. Abgänge in dem betreffenden Raum speichern, zum andern muß er eine der 20 verschiedenen Objekte und Gefahren anzeigen. Diese 28 Situationen stehen nun den 16 Bits einer Zahl gegenüber, so daß offensichtlich wird, daß nicht ein Bit eine Situation anzeigt.
Ich habe mich hier für die folgende Lösung entschieden: Die möglichen Ausgänge eines Raumes erhalten jeweils ein eigenes Bit, welches ihre Existenz anzeigt, wohingegen eine der 20 Situationen durch die Kombination der Bits 8 bis 12 gespeichert ist. Die Tabelle zeigt, welche Bits welche Objekte und Türen initialisieren.

NUR WENIGE ZEILEN FÜR VIELE OBJEKTE

Für – im Rahmen der Möglichkeiten des TI – beliebig viele Objekte werden in dieser Weise zur Erkennung nur wenige Programmzeilen benötigt. Mittels einer Schleife 0 bis 6 wird bei der Erkennung, welche Tür im Raum existiert, die entsprechende Zweierpotenz mit dem Array-Inhalt verglichen und bei Übereinstimmung in ein Unterprogramm verzweigt.
Wie oben erwähnt, wählte ich für das Erkennen der Gegenstände eine Kombination der Bits 8 bis 12. Das Bit 8 entspricht dem dezimalen Wert 128. Eine weitere Addition von 128 bewirkt das Anschalten des 9. bzw. Löschen des 8. Bits. In dieser Weise entsteht bei jeder Addition ein kennzeichnendes Bitmuster.
Die Objekt- u. Gegenstandroutine überprüft nun anhand einer Schleife von 20 bis 1 abwärts, ob das Bitmuster des entsprechenden Produktes mit 128

Bitte lesen Sie weiter auf Seite 20

```
10 ! ***********************
11 ! *   WIZARD  CASTLE    *
12 ! *      (Teil 1)       *
13 ! *                     *
14 ! *    Copyright by     *
15 ! *                     *
16 ! *  Peter  Krawinkel   *
17 ! *                     *
19 ! * Benoetigte Geraete  *
20 ! *   TI99/4A  Konsole  *
21 ! *     Ext. Basic      *
22 ! *   Cassettenrec.     *
23 ! *                     *
26 ! *  Speicherbelegung   *
27 ! *     3583 Bytes      *
28 ! *                     *
29 ! ***********************
100 CALL CLEAR :: CALL SCREE
N(5):: FOR I=0 TO 14 :: CALL
 COLOR(I,16,1):: NEXT I :: C
ALL COLOR(1,15,5)
110 !
120 ! CHAR-DEFINITIONEN
130 !
140 CALL CHAR(34,"0F1F3F7F7F
7F7F3FF0F8FCFEFEFEFEFEFC")
150 CALL CHAR(36,"060F0F0703
"&RPT$("01",11)&"60F0F0E0C0"
&RPT$("80",11))
160 CALL CHAR(40,RPT$("00FBF
BFB00DFDFDF",2)&RPT$("00F6F6
F600BEBEBE",2))
170 CALL CHAR(44,RPT$("FF040
404FF202020",2)&RPT$("FE0808
08FE404040",2))
180 CALL CHAR(48,"0000000001
0101070F0F1F1F0F0F0701003840
80808080E0F0F0F8F8F0F0E08")
190 CALL CHAR(52,"3F40FFFFFF
FFC3EDEDE3EDEDC3FFFFFFF030"&
RPT$("D0",11)&"E0C0C")
200 CALL CHAR(56,"0708080701
01031F3F7FFF88BAAA88FFE01010
E08080C0F8FCFEFFB3B5B593FF")
210 CALL CHAR(60,"11191D1F1F
1F1D1911"&RPT$("01",7)&"8080
80E0E0E0"&RPT$("80",10))
220 CALL CHAR(64,"3C4299A1A1
99423C")
230 CALL CHAR(65,"387CEEC6FE
FEC6C6FCFEC6FCFCC6FEFC7CFEC0
C0C0C0FE7CFCFEC6C6C6C6FEFC")
240 CALL CHAR(69,"FEFEC0F8F8
C0FEFEFEFEC0F8F8C0C0C07CFEC0
CECEC6FE7CC6C6C6FEFEC6C6C6")
250 CALL CHAR(73,RPT$("18",8
)&"FEFE06060606FE7CC6CEDCF8F
8DCCEC6C0C0C0C0C0C0FEFE")
```

LISTING

```
260 CALL CHAR(77,"C6EEFEFED6
C6C6C6C6E6E6F6DECECEC67CFEC6
C6C6C6FE7CFCFEC6FEFCC0C0C0")
270 CALL CHAR(81,"3C7E666666
6E3C1AFCFEC6FEFCDCCEC67CFEC0
FC7E06FE7CFEFE181818181818")
280 CALL CHAR(85,"C6C6C6C6C6
C6FE7CC6C6EE6C7C383810C6C6C6
C6D6FEFE6C6666663C3C666666")
290 CALL CHAR(89,"6666663C18
1818187E7E060C18307E7E")
300 PRINT TAB(7);"WIZARD CAS
TLE @": : :TAB(5);"EIN ADVEN
TURE GAME": :TAB(12);"VON":
:TAB(7);"PETER KRAWINKEL": :
 : : : : :
310 PRINT "SIE HABEN SOEBEN
DEN ": :"ZEICHENGENERATOR GE
STARTET": : :"TASTE L..FUER
SPIEL LADEN": :"TASTE I FUER
 INFORMATIONEN"
320 CALL KEY(0,K,S):: IF S=0
 THEN 320 ELSE IF K=76 THEN
330 ELSE IF K=73 THEN GOSUB
500 ELSE 320
330 CALL CHAR(92,"0E1F377F1F
4E3E06070F0F4F3F0F1F3F000080
80000C12121420A0A0A0A0A0C")
340 CALL CHAR(96,"030F1C3367
6CCBDCDFCB6C67331C0F03C0F038
CCE636D3FB3BD336E6CC38F0C")
350 CALL CHAR(100,RPT$("0",1
5)&"102040A19241008000038444
4245880000000000000000")
360 CALL CHAR(104,RPT$("0",1
3)&"8182F7F0F030303070E"&RPT
$("0",16)&"F8FCFEFE8F870307"
)
370 CALL CHAR(108,"000103040
6040302040604030606261CE0F0C
848C040808040C04080C0C0C87")
380 CALL CHAR(112,"80603C1F0
F1F7C3B0B0D0E1F1930200040E0E
0F8FF7EB8D8BCDE59B8FCFC8E06"
)
390 CALL CHAR(116,RPT$("01",
16)&RPT$("80",16))
400 CALL CHAR(120,"010303060
2020202020100003FC80FC02F0F81
8AC48480848A810A0F00C4A4AEA"
)
410 CALL CHAR(124,"020202020
3"&RPT$("02",9)&"060F4A4A4A0
AFC080848A8A8A8A8A8A8ACBE")
420 CALL CHAR(128,RPT$("01",
8)&"FF"&RPT$("0",14)&"804020
1008040201010204081020408")
430 CALL CHAR(132,"FF0204081
0204080FF4020100804020181828
48890A0C0808141211109050301"
)
440 CALL CHAR(136,"010305091
1214181FF010101010101FF"&RPT
$("80",7)&"FFFF"&RPT$("01",7
))
450 CALL CHAR(140,"FF"&RPT$(
"0",12)&"FFFF03050911214181F
F41211109050301007E242466242
466")
460 CALL CLEAR :: RUN "CS1"
470 !
480 ! SPIELERLAEUTERUNG
490 !
500 CALL CLEAR :: PRINT "WIZ
ARD CASTLE": : :"IST EIN GEM
AEUER VOLLER GE~": :"FAHREN
UND WILDER ABENTEUER": : :
510 PRINT "MAN VERSCHLEPPTE
SIE HIERHIN": :"ALS SIE DIE
TOCHTER DES KOE~": :"NIGS BE
FREIEN WOLLTEN": : :
520 PRINT "NUN LIEGT ES GANZ
 AN IHNEN": :"OB SIE WIZARD
CASTLE JEMALS": :"LEBEND WIE
DER VERLASSEN !!"
530 CALL KEY(0,K,S):: IF S=0
 THEN 530 ELSE CALL CLEAR
540 PRINT "ZU IHRER UNTERSTU
ETZUNG SEI": :"FOLGENDES GES
AGT": : :"MIT N~O~S~W GEHEN
SIE IN DIE": :"VIER HIMMELSR
ICHTUNGEN": :
550 PRINT "MIT H ODER R KLET
TERN SIE": :"EINE LEITER HOC
H ODER RUNTER": : :"DIE VON
IHNEN VERLANGTEN AN~": :
560 PRINT "WEISUNGEN GEBEN S
IE IN DER": :"FORM..VERB NOM
EN..EIN....ZUM": :"BEISPIEL.
..VERBRENNE BUCH": : :
570 CALL KEY(0,K,S):: IF S=0
 THEN 570 ELSE CALL CLEAR
580 PRINT "SEIEN SIE AUF DER
 HUT !!!": : : :"WIZARD CAST
LE BIRGT VIELE": :"RAETSEL I
N SICH !!!!!!!!!!": : : : :
: : :
590 CALL KEY(0,K,S):: IF S=0
 THEN 590 ELSE CALL CLEAR ::
 RETURN
```

dem Bitmuster des Array-Inhaltes entspricht. Bei Übereinkunft wird ebenfalls in die entsprechenden Unterprogramme verzweigt.

Erkennungsroutine "nehmen":
Das Array C$ enthält eine Tabelle aller möglichen Gegenstände und Objekte. Nach dem Erkennen des

Schlüsselwortes "nimm" in Zeile 430 wird in den Zeilen 1650 – 1750 der Teilstring nach "nimm" auf die Strings in der Tabelle C$ überprüft. Besteht Übereinstimmung, wird nun die Existenz des Gegenstandes im Raum verifiziert. Trägt der Spieler bereits zwei Objekte, kann er kein weiteres aufnehmen, trägt er nur eins, kann es unter Umständen zu unmöglichen Kombinationen kommen. Diese sind zum Beispiel Buch-Fackel oder Bombe-Fackel. Dies veranlaßt die Verzweigung zu einer kurzen Meldung.

LANGER SPIELVERLAUF GESICHERT

Andernfalls nimmt der Spieler den Gegenstand und trägt ihn solange, bis er ihn braucht, verliert oder geraubt wird.
– Zeilen 1650 – 1750 – Erkennungsroutine "verlieren":
Ähnlich wie in der Routine "nehmen" wird in der Routine "verlieren" der Befehlsstring auf ein in der Tabelle enthaltenes Wort überprüft. Ist dies der Fall, wird der Gegenstand im Raum abgelegt, sofern der Raum vorher leer war. Das Ablegen erfolgt mittels Sprung in die entsprechende Zeile der Objektdarstellung.
– Zeilen 1150 – 1230 – Reaktionen auf Spielerkommandos:
Die möglichen Antworten bzw. Befehle, die der Spieler bei Erkennen einer Gefahr geben kann, sind in der Tabelle C$ abgelegt. Hier sind neun Antwortstrings festgelegt. Diese Methode der Speicherung bietet den Vorteil, daß der abtippende Spieler den Verlauf des Spieles weniger leicht durchschaut. Der jedesmal neu gebildete Zauberspruch setzt darüber hinaus voraus, daß immer jedes Stockwerk durchsucht werden muß und somit ein langer Spielverlauf beschert bleibt.

TABELLE DER MÖGLICHEN INFORMATIONEN

Bitwert	Bit 12	11	10	9	8	7	6	5	4	3	2	1	Information
1												|	Grundmauern
2											|		Tür – Osten
4										|			Tür – Westen
8									|				Tür – Norden
16								|					Tür – Süden
32							|						Aufstieg
64						|							Abstieg
128					|								Schlüssel
256				|									Bombe
384				|	|								Fleisch
512			|										Schild
640			|		|								Beil
768			|	|									Fackel
896			|	|	|								Gold
1024		|											Tarnkappe
1152		|			|								Kobold rechts
1280		|		|									Kobold links
1408		|		|	|								Mauer
1536		|	|										Einzelwache
1664		|	|		|								Doppelwache
1792		|	|	|									Drache
1920		|	|	|	|								Bär
2048	|												Dunkelheit
2176	|				|								Truhe
2304	|			|									Ausgang
2432	|			|	|								Buch
2560	|		|										Schriftzeichen

```
10 ! ************************
11 ! *   WIZARD  CASTLE    *
12 ! *      (Teil 2)       *
13 ! *                     *
14 ! *    Copyright by     *
15 ! *                     *
16 ! * Peter  Krawinkel    *
17 ! *                     *
19 ! * Benoetigte Geraete  *
20 ! *  TI99/4A  Konsole   *
21 ! *     Ext. Basic      *
22 ! *    Cassettenrec.    *
23 ! *                     *
26 ! *  Speicherbelegung   *
27 ! *     11746 Bytes     *
28 ! *                     *
29 ! ************************
100 CALL CLEAR :: CALL SCREE
N(5):: ON WARNING NEXT :: RA
NDOMIZE
110 OPTION BASE 1 :: DIM R(4
,4,4),B$(7),C$(11),D$(8),F$(
5),G$(9):: MC$="OWNSRH" :: G
G,AG=0 :: A=128
120 DEF ZF(X)=INT(RND*X)+1 :
: DEF ZG=INT(RND*5)+11 :: DE
F ZH=INT(RND*2)+2
130 RESTORE 1880 :: FOR I=1
TO 7 :: FOR J=1 TO 5 :: READ
 CO :: B$(I)=B$(I)&CHR$(CO):
: NEXT J :: NEXT I
140 FOR I=1 TO 11 :: READ C$
(I):: NEXT I :: FOR I=1 TO 8
 :: READ D$(I):: NEXT I :: F
OR I=1 TO 5 :: READ F$(I)::
NEXT I
150 ZA$="" :: FOR I=1 TO 6 :
: L=ZF(5):: ZA$=ZA$&F$(L)::
NEXT I :: RESTORE 1930
160 FOR I=1 TO 9 :: READ M,N
 :: G$(I)=D$(M)&C$(N):: NEXT
 I
170 !
180 ! RAEUME
190 !
200 FOR K=1 TO 4 :: FOR J=1
TO 4 :: FOR I=1 TO 4 :: R(K,
J,I)=31
210 IF I=1 THEN R(K,J,I)=R(K
,J,I)AND 29 ELSE IF I=4 THEN
 R(K,J,I)=R(K,J,I)AND 27
220 IF J=1 THEN R(K,J,I)=R(K
,J,I)AND 15 ELSE IF J=4 THEN
 R(K,J,I)=R(K,J,I)AND 23
230 NEXT I :: NEXT J :: NEXT
 K
240 N=ZH :: FOR I=1 TO 4 ::
READ Y,Z :: R(4,4,N+Y)=R(4,4
,N+Y)+Z :: NEXT I :: R(4,3,N
)=R(4,3,N)+A*ZG :: FOR I=1 T
O 3
250 L=ZF(4):: M=ZH :: N=ZH :
: IF R(L,M,N)>A THEN 250 ELS
```

LISTINGS

```
E 260
260 FOR J=1 TO 5 :: READ X,Y
,Z :: R(L,M+X,N+Y)=R(L,M+X,N
+Y)+Z :: NEXT J :: R(L,M-1,N
)=R(L,M-1,N)+A*ZG :: NEXT I
270 FOR K=2 TO 4
280 M=ZF(4):: N=ZF(4):: IF R
(K,M,N)>A OR R(K-1,M,N)>A TH
EN 280 ELSE R(K,M,N)=R(K,M,N
)+32 :: R(K-1,M,N)=R(K-1,M,N
)+64
290 NEXT K :: FOR I=2 TO 9 :
: READ X :: FOR J=1 TO X
300 L=ZF(3):: M=ZF(4):: N=ZF
(4):: IF R(L,M,N)>A THEN 300
 ELSE R(L,M,N)=R(L,M,N)+I*12
8
310 NEXT J :: NEXT I :: FOR
I=10 TO 16 :: READ X,Y :: FO
R J=1 TO X
320 L=INT(RND*3)+2 :: M=ZF(4
):: N=ZF(4):: IF R(L,M,N)>A
OR Y*M=8 OR Y*N=4 THEN 320 E
LSE R(L,M,N)=R(L,M,N)+I*A
330 IF Y=0 THEN 340 ELSE IF
Y=1 AND N<4 THEN R(L,M,N+1)=
R(L,M,N+1)-2 :: R(L,M,N)=R(L
,M,N)-4
340 IF Y=2 AND M<4 THEN R(L,
M+1,N)=R(L,M+1,N)-16
350 NEXT J :: NEXT I :: Z,Y,
X=1 :: GOTO 470
360 !
370 ! BEWEGUNGSROUTINE
380 !
390 CALL AC(W$):: IF LEN(W$)
=1 THEN 400 ELSE IF SEG$(W$,
1,9)=D$(6)THEN 1110 ELSE IF
SEG$(W$,1,5)=D$(1)THEN 1610
ELSE 1170
400 P=POS(MC$,W$,1):: IF P=0
 THEN 390 ELSE IF R(Z,Y,X)AN
D 2^P THEN ON P GOTO 410,410
,420,420,430,430 ELSE 390
410 X=X+(-1)^P :: GOTO 470
420 Y=Y-(-1)^P :: GOTO 470
430 Z=Z+(-1)^P
440 !
450 ! TUERENROUTINE
460 !
470 IF INT(R(Z,Y,X)/A)=16 TH
EN CALL COLOR(13,5,5,14,5,5)
480 FOR L=0 TO 6 :: IF R(Z,Y
,X)AND 2^L THEN 490 ELSE 500
490 ON L+1 GOSUB 630,670,700
,720,760,770,800
500 NEXT L
510 PS=R(Z,Y,X)AND 72 :: IF
PS=72 THEN CALL HCHAR(10,16,
139):: CALL HCHAR(10,14,142)
520 !
530 ! OBJEKTROUTINE
540 !
```

LISTINGS

```
550 IF R(Z,Y,X)<A THEN 390
560 FOR L=20 TO 1 STEP -1 ::
 PS=R(Z,Y,X)AND A*L :: IF PS
=L*A THEN 570 ELSE 590
570 IF L>10 THEN 580 ELSE ON
 L GOTO 880,890,900,910,920,
930,940,950,960,970
580 ON L-10 GOTO 980,990,100
0,1020,1030,1440,1040,1050,1
060,1070
590 NEXT L :: GOTO 390
600 !
610 ! RAEUME PRINTEN
620 !
630 CALL CLEAR :: CALL DELSP
RITE(ALL):: FOR I=1 TO 6 ::
CALL HCHAR(I,I,130):: CALL H
CHAR(25-I,33-I,130)
640 CALL HCHAR(I,33-I,131)::
 CALL HCHAR(25-I,I,131):: NE
XT I :: CALL HCHAR(7,7,129,2
0)
650 CALL HCHAR(19,7,129,20):
: CALL VCHAR(7,6,128,12):: C
ALL VCHAR(8,26,128,11)
660 CALL HCHAR(7,26,139):: R
ETURN
670 CALL VCHAR(10,28,128,10)
:: CALL VCHAR(9,30,128,14)
680 CALL HCHAR(9,29,131):: C
ALL HCHAR(8,30,136):: CALL H
CHAR(10,29,129):: CALL HCHAR
(10,30,139)
690 CALL HCHAR(20,28,135)::
CALL HCHAR(21,29,129):: CALL
 HCHAR(21,30,139):: RETURN
700 CALL VCHAR(8,2,128,15)::
 CALL VCHAR(11,4,128,10):: C
ALL HCHAR(8,3,130):: CALL HC
HAR(9,4,130)
710 CALL HCHAR(10,3,129):: C
ALL HCHAR(10,4,139):: CALL H
CHAR(21,3,129,2):: CALL HCHA
R(22,3,32):: RETURN
720 CALL VCHAR(10,13,128,9):
: CALL VCHAR(11,17,128,8)::
CALL HCHAR(10,14,129,3)
730 CALL HCHAR(10,17,141)::
CALL HCHAR(19,14,32,4):: CAL
L VCHAR(11,14,128,7):: CALL
VCHAR(11,16,128,7)
740 CALL HCHAR(10,14,133)::
CALL HCHAR(18,14,131):: CALL
 HCHAR(18,17,135)
750 CALL HCHAR(11,15,129)::
CALL HCHAR(11,16,139)
760 RETURN
770 CALL HCHAR(20,14,129,3):
: CALL HCHAR(21,14,129,4)::
CALL HCHAR(20,17,139)
780 CALL HCHAR(20,13,136)::
CALL HCHAR(20,18,130):: FOR
I=0 TO 1 :: CALL HCHAR(21,12
+I*6,131-I,2):: NEXT I
790 CALL HCHAR(22,12,129,8):
: RETURN
800 FOR I=0 TO 1 :: CALL HCH
AR(2,12+I*6,133-I,2):: NEXT
I :: CALL HCHAR(2,14,129,4)
810 CALL HCHAR(3,13,135):: C
ALL HCHAR(3,18,134):: CALL H
CHAR(3,14,137):: CALL VCHAR(
3,15,140)
820 CALL HCHAR(3,16,137):: C
ALL HCHAR(3,17,140):: FOR I=
5 TO 17 STEP 2 :: CALL HCHAR
(I,15,129)
830 CALL HCHAR(I,16,139):: N
EXT I :: CALL VCHAR(4,14,128
,14):: CALL HCHAR(4,16,128)
840 CALL HCHAR(7,14,139):: F
OR I=6 TO 16 STEP 2 :: CALL
HCHAR(I,16,128):: NEXT I ::
RETURN
850 !
860 ! GEGENSTAENDE
870 !
880 CALL VG(3):: CALL SPRITE
(#1,100,16,100,33):: GOTO 39
0
890 CALL VG(3):: CALL SPRITE
(#1,48,2,141,198):: GOTO 390
900 CALL VG(3):: CALL SPRITE
(#1,36,16,137,177):: CALL VC
HAR(20,23,34,2):: CALL VCHAR
(20,24,35,2):: GOTO 390
910 CALL VG(4):: CALL SPRITE
(#1,96,12,95,60):: GOTO 390
920 CALL VG(3):: CALL SPRITE
(#1,60,2,90,208,#2,116,2,98,
208):: GOTO 390
930 CALL VG(3):: CALL SPRITE
(#1,112,12,90,55,#2,116,2,10
4,56):: GOTO 390
940 CALL VG(3):: CALL SPRITE
(#1,56,12,135,50):: GOTO 390
950 CALL VG(2):: CALL SPRITE
(#1,84,8,89,161):: GOTO 390
960 CALL VG(4):: CALL SPRITE
(#1,108,15,138,213):: GOTO 3
90
970 CALL VG(4):: CALL SPRITE
(#1,108,15,138,14):: GOTO 11
80
980 CALL VG(4):: CALL SPRITE
(#2,40,10,114,105,#3,44,15,1
14,105):: GOTO 1180
990 CALL VG(4):: CALL SPRITE
(#1,120,16,87,101,#2,124,16,
119,101,#3,60,2,86,82,#4,116
,2,117,82):: GOTO 1180
1000 CALL VG(4):: CALL SPRIT
E(#1,120,15,87,150,#2,124,15
,119,150,#3,60,2,87,132,#4,1
16,2,119,132)
1010 CALL SPRITE(#5,120,15,8
```

```
7,73,#6,124,15,119,73,#7,60,
2,87,55,#8,116,2,119,55):: G
OTO 1180
1020 CALL VG(4):: CALL HCHAR
(18,15,34):: CALL HCHAR(18,1
6,35):: CALL SPRITE(#1,92,15
,108,109):: GOTO 1370
1030 CALL VG(4):: CALL SPRIT
E(#1,104,14,117,107):: GOTO
1180
1040 FOR I=4 TO 7 :: DISPLAY
 AT(13+I,6)SIZE(5):B$(I):: N
EXT I :: GOTO 390
1050 DISPLAY AT(10,10)SIZE(7
):"AUSGANG" :: FOR I=110 TO
550 STEP 10 :: CALL SOUND(-1
39,I,5):: NEXT I :: GOTO 390
1060 CALL VG(3):: CALL SPRIT
E(#1,52,15,125,65):: GOTO 39
0
1070 FOR J=1 TO 12 :: CALL H
CHAR(13,J+9,INT(RND*4)+140):
: NEXT J :: GOTO 390
1080 !
1090 ! VERLIEREN
1100 !
1110 FOR I=1 TO 10 :: IF SEG
$(W$,10,LEN(W$)-9)=C$(I)THEN
 1140 ELSE 1120
1120 NEXT I
1130 CALL ME(24,"DAS IST NIC
HT MOEGLICH !"):: CALL PA(35
0):: GOTO 390
1140 IF R(Z,Y,X)>A THEN 1130
 ELSE IF GG AND 2^I THEN CAL
L GT(AG,GG,I):: R(Z,Y,X)=R(Z
,Y,X)+A*(I+(-9*(I=10))):: GO
TO 1150 ELSE 1130
1150 ON I GOTO 880,890,900,9
10,920,930,940,950,960,1060
1160 GOTO 390
1170 PS=INT(R(Z,Y,X)/A)-16 :
: IF PS<1 OR PS>4 THEN 390 E
LSE ON PS GOTO 1490,1540,155
0,1560 ELSE 390
1180 CALL AC(W$):: IF W$<>"N
" AND LEN(W$)=1 THEN 1190 EL
SE ON INT(R(Z,Y,X)/A)-9 GOTO
 1230,1270,1310,1340,1370,14
10,1440
1190 P=POS(MC$,W$,1):: IF P=
0 THEN 1180 ELSE IF R(Z,Y,X)
AND 2^P THEN ON P GOTO 410,4
10,420,420,430,430 ELSE 1180
1200 !
1210 ! GEFAHREN
1220 !
1230 IF W$=G$(1)AND(GG AND 2
56)THEN 1240 ELSE IF W$="W"
THEN 1180 ELSE IF W$="N" THE
N 1190 ELSE 1180
1240 CALL ME(24,"DU BIST NUN
 UNSICHTBAR !"):: CALL PA(50
```

LISTINGS

```
0)
1250 CALL AC(W$):: IF W$="W"
 THEN 1260 ELSE IF LEN(W$)=1
 THEN CALL GT(AG,GG,8):: GOT
O 400 ELSE 1250
1260 R(Z,Y,X)=R(Z,Y,X)-1276
:: X=X+1 :: CALL GT(AG,GG,8)
:: R(Z,Y,X)=R(Z,Y,X)+2 :: GO
TO 470
1270 IF W$=G$(2)AND(GG AND 4
)THEN CALL SPRITE(#1,48,2,13
0,115):: CALL PA(250):: CALL
 PATTERN(#1,112)ELSE 1180
1280 CALL COLOR(#1,12):: CAL
L SOUND(339,-7,0):: CALL SPR
ITE(#4,112,11,112,90):: CALL
 SOUND(339,-6,0):: CALL PATT
ERN(#2,112,#3,112)
1290 CALL DELSPRITE(#1,#2,#3
):: CALL SOUND(339,-7,0):: C
ALL DELSPRITE(ALL):: R(Z,Y+1
,X)=R(Z,Y+1,X)+16
1300 R(Z,Y,X)=R(Z,Y,X)-1408
:: CALL GT(AG,GG,2):: GOTO 3
90
1310 W1$=D$(7)&"WACHE" :: IF
 W$=W1$ AND(GG AND 32)THEN 1
320 ELSE 1450
1320 CALL ME(24,"DU WARST SE
HR TAPFER !"):: CALL PA(500)
:: CALL DELSPRITE(ALL)
1330 R(Z,Y,X)=R(Z,Y,X)-1536
:: R(Z,Y+1,X)=R(Z,Y+1,X)+16
:: CALL GT(AG,GG,5):: GOTO 3
90
1340 IF W$=G$(3)AND(GG AND A
)THEN 1350 ELSE 1800
1350 CALL ME(23,"DAS IST ES
WAS DIESE GELDGIERIGEN BRUED
ER BRAUCHEN !"):: CALL PA(95
0):: CALL GT(AG,GG,7)
1360 R(Z,Y,X)=R(Z,Y,X)-1664
:: CALL DELSPRITE(ALL):: CAL
L HCHAR(23,3,32,28):: R(Z,Y+
1,X)=R(Z,Y+1,X)+16 :: GOTO 3
90
1370 CALL AC(W$):: IF W$=G$(
4)AND(GG AND 16)OR W$=G$(5)A
ND(GG AND 512)THEN 1380 ELSE
 1820
1380 CALL SPRITE(#2,96-12*(W
$=W2$),11,108,35,#3,112,12,1
08,100):: FOR I=90 TO 35 STE
P -3 :: CALL LOCATE(#3,108,I
)
1390 CALL SOUND(-139,-6,5)::
 NEXT I :: FOR I=35 TO 108 S
TEP 3 :: CALL LOCATE(#3,108,
I):: CALL SOUND(-139,-5,3)::
 NEXT I :: CALL DELSPRITE(AL
L)
1400 R(Z,Y,X)=R(Z,Y,X)-1792
:: CALL GT(AG,GG,(9+5*(W$=G$
```

LISTINGS

```
(4))):: R(Z,Y+1,X)=R(Z,Y+1,
X)+16 :: CALL HCHAR(18,15,32
,2):: GOTO 390
1410 IF W$=G$(6)AND(GG AND 8
)OR W$=G$(5)AND(GG AND 512)T
HEN 1420 ELSE 1810
1420 R(Z,Y,X)=R(Z,Y,X)-1920
:: CALL GT(AG,GG,(9+6*(W$=G$
(6)))):: CALL ME(24,"DER BAE
R IST JETZT SATT !")
1430 CALL PA(750):: CALL DEL
SPRITE(ALL):: R(Z,Y+1,X)=R(Z
,Y+1,X)+16 :: GOTO 390
1440 CALL AC(W$):: IF W$=G$(
7)AND(GG AND 64)OR W$=G$(3)A
ND(GG AND 128)THEN 1470 ELSE
 1450
1450 CALL CLEAR :: CALL ME(2
0,"DIE WAECHTER RAUBEN DICH
AUSUND VERSCHLEPPEN DICH IN
DIETIEFSTEN VERLIESE !")
1460 CALL PA(1100):: AG,GG=0
 :: Z=ZF(2):: Y=ZF(4):: X=ZF
(4):: CALL COLOR(13,16,1,14,
16,1):: GOTO 470
1470 R(Z,Y,X)=R(Z,Y,X)-2048
:: CALL GT(AG,GG,(7+(W$=G$(7
)))):: IF W$=G$(7)THEN E$="U
ND ES WARD LICHT !" ELSE E$=
"DIE RAEUBER ZIEHEN AB !"
1480 CALL ME(24,E$):: CALL P
A(350):: CALL COLOR(13,16,1,
14,16,1):: GOTO 390
1490 IF W$=G$(8)AND(GG AND 2
)THEN 1500 ELSE 390
1500 FOR I=1 TO 3 :: DISPLAY
 AT(14+I,6)SIZE(5):B$(I):: N
EXT I :: CALL VG(3):: CALL S
PRITE(#1,52,15,115,76):: CAL
L GT(AG,GG,1)
1510 CALL AC(W$):: IF W$=G$(
9)THEN 1520 ELSE IF LEN(W$)=
1 THEN 1530
1520 IF AG=2 THEN 1510 ELSE
AG=AG+1 :: GG=GG+1024 :: GOT
O 390
1530 R(Z,Y,X)=R(Z,Y,X)+256 :
: GOTO 400
1540 IF W$<>ZA$ THEN CALL ME
(24,"DAS WAR WOHL NICHTS !")
:: CALL PA(650):: GOTO 390 E
LSE 1720
1550 IF W$=G$(9)AND AG<2 THE
N AG=AG+1 :: GG=GG+1024 :: R
(Z,Y,X)=R(Z,Y,X)-2432 :: GOT
O 390 ELSE 390
1560 IF W$=D$(8)AND(GG AND 1
024)THEN 1570 ELSE 390
1570 DISPLAY AT(13,8)SIZE(12
)BEEP:ZA$ :: R(Z,Y,X)=R(Z,Y,
X)-2560 :: CALL GT(AG,GG,10)
:: GOTO 390
1580 !
1590 ! NEHMEN
1600 !
1610 FOR I=1 TO 10 :: IF SEG
$(W$,6,LEN(W$)-5)=C$(I)THEN
1640 ELSE 1620
1620 NEXT I
1630 CALL ME(24,"ICH SEHE SO
ETWAS NICHT !"):: CALL PA(35
0):: GOTO 390
1640 PS=R(Z,Y,X)AND A*(I+(-9
*(I=10))):: IF AG=2 OR(GG AN
D 2^I)THEN 390 ELSE IF PS=A*
(I+(-9*(I=10)))THEN 1650 ELS
E 1630
1650 IF AG=0 THEN 1680 ELSE
IF AG=2 THEN 400
1660 RESTORE 1970 :: FOR J=1
 TO 4 :: READ M :: IF GG+2^I
=M THEN ON J GOTO 1790,1730,
1740,1750,1760 ELSE 1670
1670 NEXT J
1680 CALL DELSPRITE(ALL):: R
(Z,Y,X)=R(Z,Y,X)-A*(I+(-9*(I
=10))):: GG=GG+2^I :: AG=AG+
1 :: CALL HCHAR(20,23,32,2):
: CALL HCHAR(21,23,32,2):: G
OTO 390
1690 !
1700 ! MELDUNGEN
1710 !
1720 E$="HURRA DU HAST ES GE
SCHAFFT..LEBEND ZU ENTKOMMEN
 ! TOLL !" :: GOTO 1830
1730 E$="DIE FACKEL VERBRANN
TE DEN...KOBOLD ! WELCH EIN
GROSSER..VERLUST !" :: GOTO
1770
1740 E$="DER KOBOLD IST MIT
DEM GOLD ABGEHAUEN ! NUN HAS
T DU.....NICHTS MEHR !" :: G
OTO 1770
1750 E$="DIE FACKEL HAT DEIN
 BUCH....VERBRANNT ! PECH FU
ER DICH !" :: GOTO 1770
1760 E$="DER KOBOLD KLAUTE D
IE.......TARNKAPPE ! ENTSCHW
UNDEN ISTER NUN !" :: GOTO 1
770
1770 CALL DELSPRITE(ALL):: C
ALL CLEAR :: CALL ME(12,E$):
: CALL PA(1500):: R(Z,Y,X)=R
(Z,Y,X)-I*A
1780 AG,GG=0 :: GOTO 470
1790 E$="DIE FACKEL ENTZUEND
ETE DIE..BOMBE ! DU BIST DAB
EI DRAUF GEGANGEN!" :: GOTO
1830
1800 E$="MIT DER WACHE LAESS
T SICH...NICHT SPASSEN ! DU
HAST ES..WOHL GEMERKT !" ::
GOTO 1830
1810 E$="DU HAST DEIN BESTES
 GEGEBEN DOCH DER BAER FRASS
```

```
DICH TROTZDEM" :: GOTO 1830
1820 E$="DER TOEDLICHE ATEM
DES......DRACHEN HAT DICH GE
TROFFEN !"
1830 CALL DELSPRITE(ALL):: C
ALL CLEAR :: CALL ME(12,E$):
: CALL PA(1500):: CALL ME(22
,"NOCH EIN SPIEL J N")
1840 CALL AC(W$):: IF W$="N"
 OR W$="n" THEN CALL CLEAR :
: END ELSE AG,GG=0 :: GOTO 1
50
1850 !
1860 ! DATEN ZUM SPIEL
1870 !
1880 DATA 32,128,129,129,139
,32,128,32,32,128,32,131,129
,129,141,32,131,132,132,141
1890 DATA 128,129,129,139,13
6,139,32,64,128,131,32,129,1
29,129,32
1900 DATA SCHLUESSEL,BOMBE,F
LEISCH,SCHILD,BEIL,FACKEL,GO
LD,TARNKAPPE,KOBOLD,BUCH,TRU
HE
1910 DATA "NIMM ","OEFFNE ",
"BENUTZE ","ZUENDE ","GIB ",
"VERLIERE ","VERJAGE ",UEBER
SETZE
1920 DATA BA,LI,PE,WO,TA
1930 DATA 3,8,4,2,5,7,3,4,5,
9,5,3,3,6,2,11,1,10
1940 DATA 0,2304,0,-22,1,-2,
-1,-4,0,0,2176,0,0,-30,0,1,-
2,0,-1,1276,1,0,-16
1950 DATA 0,0,128,0,0,-30,0,
1,-2,0,-1,1276,1,0,-16,0,0,2
560,0,0,-30,0,1,-2,0,-1,1276
,1,0,-16
1960 DATA 3,3,2,3,7,4,3,2,2,
1,2,2,2,2,2,2,2,2,2,2,5,0
1970 DATA 68,576,640,1088,76
8
1980 !
1990 ! UNTERPROGRAMME
2000 !
2010 SUB PA(A):: FOR I=1 TO
A :: NEXT I :: SUBEND
2020 SUB ME(A,A$):: DISPLAY
AT(A,1)BEEP:A$ :: SUBEND
2030 SUB VG(A):: CALL MAGNIF
Y(A):: SUBEND
2040 SUB AC(A$):: ACCEPT AT(
24,1)BEEP:A$ :: SUBEND
2050 SUB GT(A,B,C):: A=A-1 :
: B=B-2^C :: SUBEND
```

BOMB

Ziel des Spiels ist es, Ede Listig, den Bombenleger, wieder ins Zuchthaus zu bringen.
Das Spiel zeichnet sich durch eine sorgfältige Gestaltung der Grafik aus. Durch Druck auf die Feuertaste von Joystick 1 gelangen Sie vom Titelbild zum Spielbild, der Teilansicht einer Burg. Sie sind ein Polizist unten an der Mauer, oben auf der Mauer befindet sich Ede. Dieser wirft mit Bomben, die von Ihnen aufgefangen und zum Entschärfen zu einer Tonne, rechts im Bild, gebracht werden müssen. Jedoch Vorsicht: Benötigen Sie dazu zu lange, wird der Zeitzünder aktiv! Wenn Sie die Bombe fallenlassen, explodiert sie ebenfalls.
Können Sie eine Bombe nicht fangen, so hängt Ihre Überlebenschance davon ab, wie weit Sie von der Explosion entfernt sind. Der Schwierigkeitsgrad steigt im Laufe des Spiels, da Ede immer schneller wird und seine Bomben immer hinterhältiger wirft.
Das Spiel ist vorzeitig zu Ende, wenn Sie drei Bomben nicht entschärfen konnten, oder Sie zu nahe an einer Explosion standen, oder die Punktzahl unter Null absinkt (da es für jede explodierte Bombe Punktabzug gibt). Steuerung: Mit dem Joystick 1 können Sie Ihre Figur in jede Richtung bewegen. Gleichzeitiges Drücken der Feuertaste verleiht doppelte Geschwindigkeit. Eine Bombe wird gefangen, indem Sie genau darunterstehen. Sie wird im Faß durch Drücken der Feuertaste abgelegt.
Falls Sie 10 Bomben gefangen haben ohne eine einzige fallen zu lassen, gibt es Bonuspunkte, und das Schlußbild erscheint.

BILDER-PUZZLE

Bei diesem Programm handelt es sich um ein Geschicklichkeitsspiel, dessen Ziel es ist, ein Bild nach dem Vermischen einzelner Bildteile durch den Computer wieder in den Originalzustand zu bringen. Das Bild besteht aus vier mal vier Segmenten, d.h., aus 15 Bildsegmenten und einer Lücke, die sich normalerweise links unten befindet. Das Wiederherstellen des Bildes erfolgt, indem man mit Hilfe des Joysticks ein Bildsegment in die Lücke hineinbewegt, wodurch wieder eine neue Lücke entsteht, in die erneut ein Segment geschoben werden kann.
Um das Bild zu mischen, verschiebt der Computer in der ersten Runde ein Bildsegment, in der zweiten Runde zwei Segmente usw.

Das Spiel gilt als beendet, wenn man zur Wiederherstellung des Bildes mehr als doppelt so viele Züge benötigt als der Computer zum Vermischen benutzte.
Der persönliche Rekord meinerseits liegt momentan bei 4012 Punkten.

LISTINGS

```
10 ! **********************
11 ! *                    *
12 ! *        BOMB        *
13 ! *                    *
14 ! *    Copyright by    *
15 ! *                    *
16 ! *  Manfred Lipowski  *
17 ! *                    *
19 ! * Benoetigte Geraete *
20 ! *  TI99/4A  Konsole  *
21 ! *     Ext. Basic     *
22 ! *     Joystick 1     *
23 ! *                    *
26 ! *  Speicherbelegung  *
27 ! *    12794 Bytes     *
28 ! *                    *
29 ! **********************
100 CALL Z :: RANDOMIZE
110 CALL CHAR(32,"",33,"FF80
8080808080FF",34,"FF01010101
0101FFFF818181818181FF",40,"
")
120 CALL COLOR(0,8,8,1,13,16
,2,7,11,3,2,16,4,2,16,5,13,1
6,14,7,11):: CALL HCHAR(1,1,
30,128)
130 FOR A=5 TO 19 STEP 2 ::
FOR B=2 TO 30 STEP 2 :: CALL
 HCHAR(A,B,33):: CALL HCHAR(
A,B+1,34):: NEXT B :: NEXT A
140 FOR A=6 TO 20 STEP 2 ::
FOR B=1 TO 31 STEP 2 :: CALL
 HCHAR(A,B,33):: CALL HCHAR(
A,B+1,34):: NEXT B :: NEXT A
150 FOR A=5 TO 19 STEP 2 ::
CALL HCHAR(A,1,35):: CALL HC
HAR(A,32,35):: NEXT A :: CAL
L HCHAR(21,1,40,128)
160 DATA 3,1,35,3,2,33,3,3,3
4,4,1,33,4,2,34,4,3,35,3,30,
33,3,31,34,3,32,35,4,30,35,4
,31,33
170 DATA 4,32,34
180 RESTORE 160 :: FOR A=1 T
O 12 :: READ B,C,D :: CALL H
CHAR(B,C,D):: NEXT A
190 CALL CHAR(48,"3C6E4A5A52
52763C0818387818181 83C3C6642
060C18307E3C66461C1C46663C")
200 CALL CHAR(52,"040C1C3464
3E04047C4040780C046C383C6440
786C446C387C04040C1830202",5
6,"3C66663C6642663C386C446C3
C046C38")
210 CALL CHAR(64,"8182848890
A0C0FFFFC0A09088848281FF0305
09112141818141211109 0503FF",
68,"8181818181818181FF000000
000000FF")
220 CALL Y(11,12,"AEEEEB")::
 CALL Y(12,12,"D0000D"):: CA
LL Y(13,12,"@EEEEC")
230 DATA 76,000000000060C19
31391E0F070703030000000000060
30988C9C78F0E0E0C0C
240 DATA 72,000000000000001
0101060F0F0F0B0B000000000000
0080808060F0F0F0D0D,88,00000
1030301,89,,90,004080C0C08,9
1,
250 DATA 80,000000000000001
010207070F0F0E0600000000000
008080800080C078,84,00000000
000000010101000 1031E00000000
0000000000808040E0E0F0F0706
260 DATA 100,030703124D23150
800000001010101020000000 80C0E
0F0F83878C8880C060204,108,00
0000103070F1F1C1E1311306040
20C0E0C048B2C4A8100000000 8080
80804
270 DATA 104,060E06259B472B1
1000103060406010000000000080C
0E0F070F060C080C0602,112,000
000000103070F0E0F06030103060
4607060A4D9E2D4880080C060206
08
280 DATA 96,01010102070F0B0B
090B0B02020202048080 8040E0F0
D0D090D0D0404040402,116,0000
000040A09043A27131C26110C0B
00000000060A02090F0201CD42850
A0C
290 DATA 120,0E0A051C150A093
44C26180D0A160B0640A0A050C02
C54A85020A090A040408,124,000
00000000501060905120509060 20
5000000000804080 40A0409050609
0204
300 DATA 128,0107050A16121D0
B0C,130,80C0A090E8B060A0C,13
2,000702060 90A07020,134,C0A0
50A040A090600
310 DATA 136,000000030C10101
C0F0F0F0F0F0F0300000000C0300
80838F0F0F0F0F0F0C,140,000
00000030F0F030,142,00000000C
0F0F0C00
320 RESTORE 230 :: FOR A=1 T
O 23 :: READ B,B$ :: CALL CH
AR(B,B$):: NEXT A
330 CALL HCHAR(22,31,136)::
CALL HCHAR(22,32,138):: CALL
 HCHAR(23,31,137):: CALL HCH
AR(23,32,139)
340 CALL SPRITE(#4,140,5,169
,241,#3,72,5,17,120,#1,96,2,
168,224):: CB,CA,CO,BA,KA,SC
=0 :: CALL Y(12,13,"      "):: 
 CALL X(SC)
350 A=INT(RND*400)+600 :: FO
R B=0 TO 30 STEP 2.5 :: CALL
 SOUND(-99,A,B,A/2,B,A+2,B):
: NEXT B :: PA=100 :: GOTO 3
70
360 CALL MOTION(#1,0,0):: CA
```



```
LL SOUND(1,-6,9):: IF KA=1 T
HEN CALL LOCATE(#1,152,Y1)EL
SE CALL LOCATE(#1,177,Y1)
370 CALL MOTION(#1,0,0):: CA
LL PATTERN(#1,96)
380 CALL JOYST(1,X,Y):: CALL
 POSITION(#1,X1,Y1,#2,X2,Y2,
#3,X3,Y3)
390 CALL COINC(ALL,C):: IF C
 THEN 890
400 IF X1<151 OR X1>178 THEN
 470 ELSE IF X2>188 THEN 500
410 IF BA<>1 AND RND<.2 THEN
 770 ELSE IF Y3<32 THEN 780
ELSE IF Y3>213 THEN 800
420 CALL COINC(ALL,C):: IF C
 THEN 890
430 IF X=-4 THEN PA=100 ELSE
 IF X=4 THEN PA=108 ELSE IF
X=0 AND Y=0 THEN 370
440 CALL KEY(1,K,S):: IF X A
ND K=18 THEN MO=4 ELSE IF Y
AND K=18 THEN MO=1.5 ELSE MO
=2
450 CALL PATTERN(#1,PA):: CA
LL SOUND(-1,-5,4):: CALL POS
ITION(#1,X1,Y1)
460 CALL COINC(ALL,C):: IF C
 THEN 890 ELSE IF X2>188 THE
N 500
470 IF X1<151 THEN KA=1 :: G
OTO 360 ELSE IF X1>178 THEN
KA=0 :: GOTO 360 ELSE CALL M
OTION(#1,-Y*MO,X*MO)
480 CALL COINC(ALL,C):: IF C
 THEN 890 ELSE IF X2>188 THE
N 500
490 CALL PATTERN(#1,PA+4)::
CALL SOUND(-1,-7,5):: GOTO 3
80
500 CALL MOTION(#1,0,0,#2,0,
0,#3,0,0):: CALL POSITION(#2
,X2,Y2):: IF Y2<2 THEN Y2=2
ELSE IF Y2>248 THEN Y2=248
510 CALL SPRITE(#8,124,7,177
,Y2,#9,124,16,176,Y2+1):: CA
=CA+1 :: BA=0 :: CALL DELSPR
ITE(#2):: CALL HCHAR(20,2,48
+CA):: FOR A=0 TO 30 STEP 10
520 CALL PATTERN(#9,124,#8,1
16):: CALL COLOR(#9,7,#8,16)
:: CALL SOUND(-99,-5,A):: CA
LL PATTERN(#9,116,#8,120)::
CALL COLOR(#9,16,#8,7)
530 CALL SOUND(-99,-6,A):: C
ALL PATTERN(#9,120,#8,116)::
 CALL COLOR(#9,10,#8,15):: C
ALL SOUND(-99,-7,A)
540 CALL PATTERN(#9,116,#8,1
24):: CALL COLOR(#9,14,#8,9)
:: CALL SOUND(-99,-6,A):: NE
XT A :: CALL DELSPRITE(#9)::
CALL COLOR(#8,2)
```

LISTINGS

```
550 CALL COINC(#8,#1,10,C)::
 IF C THEN SC=SC-100 :: CALL
 DELSPRITE(#8):: CALL SOUND(
-99,-5,9):: CALL X(SC):: GOT
O 610
560 CALL PATTERN(#8,128):: C
ALL MOTION(#8,-9,-INT(RND*8)
+4):: CALL PATTERN(#8,132)::
 CALL POSITION(#8,X3,Y3):: C
ALL SOUND(-999,-5,29)
570 IF X3>10 THEN 560 ELSE C
ALL DELSPRITE(#8):: CALL SOU
ND(-1,-5,0)
580 IF SC=0 THEN 600
590 CALL Y(12,13,"....")::  F
OR A=1 TO 10 :: SC=SC-10 ::
CALL SOUND(1,444-A*9,A):: CA
LL X(SC):: NEXT A
600 IF SC<=0 OR CA=3 THEN 61
0 ELSE 380
610 IF SC<0 THEN CALL Y(12,1
3,"...."):: CALL HCHAR(12,15
,48,4)
620 FOR A=0 TO 20 STEP 2 ::
CALL SCREEN(7):: CALL COLOR(
#1,7):: CALL SOUND(-99,-5,A)
630 CALL SCREEN(16):: CALL C
OLOR(#1,5):: CALL SOUND(-99,
-6,A):: CALL SCREEN(2):: CAL
L COLOR(#1,1):: CALL SOUND(-
99,-7,A):: NEXT A
640 FOR A=1 TO 999 :: NEXT A
 :: CALL CLEAR :: CALL DELSP
RITE(ALL):: CALL COLOR(1,1,1
):: CALL CHARSET :: FOR A=1
TO 14 :: CALL COLOR(A,13,1)
650 IF SC<0 THEN SC=0
660 NEXT A :: CALL Y(1,12,"B
 O M B"):: CALL Y(4,2,"ERREI
CHTE PUNKTZAHL:"):: DISPLAY
AT(4,22):SC
670 IF SC=0 THEN 710 ELSE IF
 SC<HI THEN 710 ELSE HI=SC :
: CALL COLOR(0,14,1)
680 CALL Y(6,2,"NEUER HIGHSC
ORE:"):: DISPLAY AT(6,18):HI
 :: CALL Y(8,2,"WER IST DER
GLUECKLICHE ?")
690 ACCEPT AT(10,2)SIZE(10)V
ALIDATE(UALPHA,". -")BEEP:NA
$ :: IF NA$="" THEN NA$="TI-
99/4A"
700 DISPLAY AT(10,2):: DISPL
AY AT(8,2):"HI-SCOREKOENIG:"
;NA$ :: DISPLAY AT(10,1)
710 DISPLAY AT(10,2):"GEFANG
ENE BOMBEN:";CB :: DISPLAY A
T(12,2):"FALLEN GELASSENE BO
MBEN:";CA
720 IF SC<HI THEN DISPLAY AT
(6,2):"ALTER HIGHSCORE:";HI
:: DISPLAY AT(8,2):"HI-SCORE
KOENIG:";NA$
```

```
730 CALL Y(14,2,"E = ENDE"):
: CALL Y(16,2,"A = ANFANG")
740 A=15 :: GOTO 760
750 CALL SOUND(-99,-5,A):: A
=A-1 :: IF A=1 THEN A=15
760 CALL KEY(0,K,S):: IF S=0
 THEN 760 ELSE IF K=65 OR K=
97 THEN 100 ELSE IF K=69 OR
K=101 THEN CALL CLEAR :: STO
P ELSE 750
770 IF BA=1 THEN 380 :: CALL
 MOTION(#1,0,X*MO):: CALL PA
TTERN(#1,PA):: CALL POSITION
(#3,X3,Y3):: ON INT(RND*3)+1
 GOTO 780,800,820
780 CALL POSITION(#3,X3,Y3):
: IF Y3<32 THEN CALL PATTERN
(#3,84):: CALL LOCATE(#3,X3,
216):: CALL MOTION(#3,0,-9):
: GOTO 380
790 CALL PATTERN(#3,84):: CA
LL MOTION(#3,0,-INT(3+CB/2))
:: GOTO 380
800 CALL POSITION(#3,X3,Y3):
: IF Y3>213 THEN CALL PATTER
N(#3,80):: CALL LOCATE(#3,X3
,40):: CALL MOTION(#3,0,9)::
 GOTO 380
810 CALL PATTERN(#3,80):: CA
LL MOTION(#3,0,INT(3+CB/2)):
: GOTO 380
820 IF CB<3 THEN FA=2 ELSE I
F CB<6 THEN FA=7 ELSE FA=16
830 CALL MOTION(#1,0,0,#3,0,
0):: CALL PATTERN(#3,76):: C
ALL SPRITE(#2,88,FA,X3,Y3)::
 CALL SOUND(1,-7,9):: BA=1
840 FOR D=1 TO 25 :: NEXT D
:: IF Y3<96 OR Y3>168 THEN C
ALL MOTION(#2,5+CB,0):: GOTO
 880
850 ON INT(RND*2)+1 GOTO 860
,870
860 CALL MOTION(#2,5+CB,-CB)
:: GOTO 880
870 CALL MOTION(#2,5+CB,CB)
880 CALL PATTERN(#3,72):: CA
LL SOUND(1,-7,5):: GOTO 380
890 CALL COINC(#1,#2,8,CO)::
 IF CO THEN 920
900 CALL COINC(#1,#4,10,C)::
 IF C THEN CALL LOCATE(#1,X1
,8):: CALL SOUND(-99,-1,8)
910 GOTO 380
920 CALL MOTION(#1,0,0,#2,0,
0,#3,0,0):: CALL SCREEN(5)
930 CALL POSITION(#1,X1,Y1):
: CALL PATTERN(#1,108):: CAL
L LOCATE(#2,X1,Y1+5):: CALL
SCREEN(16)
940 FOR A=0 TO 28 STEP 4 ::
CALL SCREEN(13):: CALL SOUND
(-99,-1,A):: CALL SCREEN(11)
:: NEXT A :: CALL SCREEN(2):
: A=22 :: BA=0
950 CALL SOUND(-333,-4,A)::
CALL JOYST(1,X,Y):: CALL POS
ITION(#1,X1,Y1,#2,X2,Y2):: A
=A-.5 :: IF A=1 THEN 500
960 CALL COINC(#2,#4,4,C)::
IF C THEN 990 ELSE IF Y1<10
THEN 980
970 CALL PATTERN(#1,108):: C
ALL MOTION(#1,-Y*1,X*2,#2,-Y
*1,X*2):: CALL POSITION(#1,X
1,Y1):: IF Y1<10 THEN 980 EL
SE CALL PATTERN(#1,112):: GO
TO 950
980 CALL MOTION(#1,0,0,#2,0,
0):: CALL SOUND(-99,-3,0)::
CALL LOCATE(#1,X1,12,#2,X2,1
7):: CALL SOUND(-99,-2,4)::
GOTO 950
990 CALL MOTION(#1,0,0,#2,0,
0):: CALL LOCATE(#1,X1,Y1-9)
:: IF CB<3 THEN LA=10 ELSE I
F CB<6 THEN LA=20 ELSE LA=30
1000 CALL SOUND(1,-5,15):: C
ALL DELSPRITE(#2):: CB=CB+1
:: CALL PATTERN(#1,96)
1010 FOR A=1 TO LA :: SC=SC+
10 :: CALL SOUND(1,810+A,A):
: CALL X(SC):: NEXT A :: IF
CB=10 THEN 1030 ELSE CALL HC
HAR(20,31,48+CB):: CALL SOUN
D(9,522+LA,5)
1020 GOTO 370
1030 IF SC<2100 THEN 640 ELS
E CALL HCHAR(20,31,33):: CAL
L SCREEN(7)
1040 FOR A=1 TO 9 :: SC=SC+1
00 :: CALL X(SC):: CALL SOUN
D(1,110*A,A,990/A,A):: NEXT
A :: FOR A=1 TO 999 :: NEXT
A
1050 CALL SCREEN(2):: CALL C
LEAR :: CALL DELSPRITE(ALL):
: CALL CHARSET
1060 CALL CHAR(40,"301C02190
10101434520101304080000018700
0300000000844408109040 2",44,"
003C4219090121434520202708l3
0C0000780430200008844408 08C8
20906")
1070 CALL CHAR(96,RPT$("F",2
0)&"7F3F3F3F1FFFFFFFFFFFFF
FEFEFEFEFCF8F8F8F8F",100,"00
000000001FFFFF0000000000F0FE
FE0101010101010100001F")
1080 CALL CHAR(104,"E",112,"
1F7FFFFFFFE"&RPT$("0",21),11
4,"F0FCFEFEFF0F0101"&RPT$("0
",16),116,"0000000001010101"
)
1090 CALL CHAR(120,"00000000
004040E0F0FFBFBFDFEFF7FB0000
```

```
00000004040E1EFFFBFBF7EFDFBF
",124,"0001013FFFFFFFFF00000
0F8FEFFFFFF")
1100 CALL CHAR(128,RPT$("07"
,16)&RPT$("0",32),132,RPT$("
E0",16)&RPT$("0",32))
1110 CALL CHAR(33,"FFFFFFFF
FFFFFFF",136,"FFFFC0C0C0C0C0
C0C0C0C0C0C0C0FFFFFFFF030303
0303030303030303FFFF")
1120 CALL COLOR(1,14,1,14,13
,16,9,15,14,10,15,14,11,5,14
,12,11,14)
1130 FOR A=1 TO 11 STEP 2 ::
 FOR B=1 TO 31 STEP 2 :: CAL
L HCHAR(A,B,136):: CALL HCHA
R(A,B+1,138):: CALL HCHAR(A+
1,B,137):: CALL HCHAR(A+1,B+
1,139)
1140 NEXT B :: NEXT A :: FOR
 A=1 TO 4 :: CALL Y(4+A,13,"
!!!!"):: NEXT A
1150 CALL Y(5,13,"tde"):: CA
LL Y(6,13,"f`b"):: CALL Y(7,
14,"ac"):: CALL Y(8,13,"!gh}
")
1160 CALL SPRITE(#6,120,11,4
9,121,#7,112,5,33,121,#5,40,
2,40,121)
1170 CALL SPRITE(#2,128,2,33
,112,#3,132,2,33,139,#1,128,
2,49,112,#4,132,2,49,139)::
CALL W
1180 CALL COLOR(2,16,1,3,8,1
,4,8,1,5,16,1,6,16,1,7,16,1,
8,16,1):: CALL Y(14,12,"B O
M B")
1190 DISPLAY AT(16,1):"HOECH
STLEISTUNG:";SC :: HI=SC ::
DISPLAY AT(18,1):"DER NAME ?
:" :: ACCEPT AT(18,12)SIZE(1
0)BEEP VALIDATE(UALPHA,". -"
):NA$
1200 IF NA$="" THEN 1190 ELS
E CALL PATTERN(#5,44)
1210 DISPLAY AT(20,1):"DU HA
ST ES GESCHAFFT EDE": :"LIST
IG SITZT HINTER GITTERN ": :
"E = ENDE..........A = ANFAN
G" :: A=15 :: GOTO 1230
1220 A=A-1 :: CALL SOUND(-99
,-6,A):: IF A=1 THEN A=15
1230 CALL KEY(0,K,S):: IF S=
0 THEN 1230 ELSE IF K=65 THE
N 100 ELSE IF K=69 THEN
1240 ELSE 1220
1240 CALL DELSPRITE(ALL):: C
ALL CLEAR :: END
1250 SUB Z :: CALL CLEAR ::
CALL DELSPRITE(ALL):: FOR A=
1 TO 24 :: PRINT :: NEXT A :
: CALL CHARSET :: CALL MAGNI
FY(3)
1260 CALL SCREEN(5):: FOR A=
1 TO 8 :: CALL COLOR(A,2,12)
:: NEXT A :: CALL COLOR(9,5,
12,10,2,12,11,16,12)
1270 CALL Y(2,3,"MANFRED LIP
OWSKI PRESENT"):: CALL CHAR(
64,"3C4299A1A199423C",96,"00
784C44444C784C44444C78")
1280 CALL CHAR(98,"00386C444
444444444446C38",100,"00446C
7C5454444444444444",102,"007
84C44444C784C44444C78")
1290 CALL CHAR(104,"0301071F
3F3F7F7F7F7F7F7F3F3F1F07C080
E0F8FCFCFEFEFEFEFEFEFCFCF8E0
",108,"000000030E18103020203
010180E03",110,"",111,"00000
0003060C")
1300 CALL CHAR(112,"08181030
2020E08",116,"",117,"",118,"
08081C1C3E3E3E3E14",120,"",1
21,"",122,"000008081C1C3E3E1
4")
1310 CALL Y(4,14,"`bdf"):: C
ALL Y(5,14,"aceg"):: CALL Y(
8,3,"@ M.LIPOWSKI NOVEMBER 8
5")
1320 CALL SPRITE(#5,96,13,31
,111,#6,100,13,31,127,#3,96,
2,38,101,#4,100,2,38,117)
1330 CALL Y(13,14,"hj"):: CA
LL Y(14,14,"ik"):: CALL HCHA
R(12,17,112)
1340 CALL SPRITE(#1,108,15,9
7,121,#2,116,7,83,121):: CAL
L Y(17,5,"PRESS FIRE TO BEGI
N")
1350 CALL SPRITE(#7,108,15,1
60,120,#8,104,2,160,120,#9,1
16,7,151,116):: A,B=0 :: GOT
O 1400
1360 A=A+1 :: IF A=2 THEN CA
LL PATTERN(#9,120,#2,116)::
A=0 ELSE CALL PATTERN(#9,116
,#2,120)
1370 CALL KEY(1,K,S):: IF K=
18 THEN 1410 ELSE B=B+1 :: I
F B=198 THEN 1410
1380 CALL POSITION(#7,X,Y,#8
,X,Y):: IF Y<20 THEN 1390 EL
SE IF Y>222 THEN 1400 ELSE C
ALL SOUND(-4250,-4,0):: GOTO
 1360
1390 CALL MOTION(#7,0,9,#8,0
,9,#9,0,9):: CALL SOUND(-99,
-7,0):: GOTO 1360
1400 CALL MOTION(#7,0,-9,#8,
0,-9,#9,0,-9):: CALL SOUND(-
99,-5,0):: GOTO 1360
1410 CALL CLEAR :: CALL COLO
R(1,2,2):: CALL DELSPRITE(AL
L):: CALL COLOR(1,1,1):: FOR
 A=0 TO 30 STEP 4 :: CALL SC
```

LISTINGS

ANZEIGENSERVICE

Die große Börse für jeden Zweck in der TI REVUE. Kostenlos für Privat-Inserenten. Sportbillig für gewerbliche Anbieter. Einfach Coupon ausschneiden, fotokopieren o.ä., ausfüllen und ab die Post – Freimachen nicht vergessen! – Unsere Adresse steht auf dem Coupon, ebenso der Preis für gewerbliche Anbieter! Achtung! Wir weisen ausdrücklich darauf hin, daß wir offensichtlich gewerbliche Anzeigen nicht kostenlos veröffentlichen und uns jedweden Abdruck kostenloser Anzeigen vorbehalten müssen, insbesondere, wenn deren Inhalt nicht TI-typisch ist oder gegen geltendes Recht verstößt. Private Chiffreanzeigen werden nicht angenommen. Für Privatanbieter: maximal acht Zeilen à 28 Anschläge. Für gewerbliche Anbieter: 5 DM p. mm.

TI REVUE
Anzeigenabteilung
Postfach 1107
8044 Lohhof

Name ____________________

Vorname ____________________

Straße/Hausnr. ____________________

PLZ/Ort ____________________

ABO-SERVICE-KARTE

COUPON

Ja, ich möchte von Ihrem Angebot Gebrauch machen.

Bitte senden Sie mir bis auf Widerruf ab sofort jeweils die nächsten zwölf Ausgaben an untenstehende Anschrift. Sollte ich nicht vier Wochen vor Ablauf schriftlich kündigen, läuft diese Abmachung automatisch weiter.

Ich nehme zur Kenntnis, daß die Belieferung erst beginnt, wenn die Abo-Gebühr dem Verlag zugegangen ist.

Name ____________________

Vorname ____________________

Straße/Hausnr. ____________________

PLZ/Ort ____________________

Ich bezahle:

☐ per beiliegendem Verrechnungsscheck

☐ gegen Rechnung

☐ bargeldlos per Bankeinzug von meinem Konto

(TI 8/86)

bei (Bank) und Ort ____________________

Kontonummer ____________________

Bankleitzahl
(steht auf jedem Kontoauszug) ____________________

Unterschrift ____________________

Von meinem Widerspruchsrecht habe ich Kenntnis genommen.

Unterschrift ____________________

TI REVUE
Abo-Service 8/86
Postfach 1107
8044 UNTERSCHLEISSHEIM

PROGRAMMSERVICE

Hiermit bestelle ich in Kenntnis Ihrer Verkaufsbedingungen
die Listings dieses Heftes auf
☐ Diskette zum Preis von (25,– DM)
☐ Kassette zum Preis von (10,– DM)

Ich zahle:
Bar – per beigefügtem Geld ( )
per beigefügtem Scheck ( )
Gegen Bankabbuchung am Versandtag ( )
Zutreffendes bitte ankreuzen!

Meine Bank (mit Ortsname) ....

Meine Kontonummer ....

Meine Bankleitzahl .... (steht auf jedem Bankauszug)

Vorname .... Nachname ....

Str./Nr. .... PLZ / Ort 8/86

Hiermit bestätige ich mit meiner Unterschrift, Ihre Verkaufsbedingungen gelesen zu haben und zu akzeptieren.

Unterschrift ....

TI-REVUE
KASSETTENSERVICE 8/86
Postfach 1107
8044 Unterschleißheim

Verkaufsbedingungen: Versand nur gegen Vorauskasse oder Bankabbuchung.
Umtauschrecht bei Nichtfunktionieren. Keine Nachnahme.

- ✂ - -

RESERVIERUNGS-SERVICE

Hiermit bestelle ich in Kenntnis Ihrer Verkaufsbedingungen

( ) Exemplar(e) TI SPECIAL (Nr. 4) 14,80 ( ) Exemplar(e) TI ASSEMBLER SPECIAL 19,80

Zutreffendes bitte ankreuzen!
Ich zahle:
per beigefügtem Scheck / Schein ( )
Gegen Bankabbuchung am Versandtag ( )
Meine Bank (mit Ortsname) 8/86

Meine Kontonummer ....

Meine Bankleitzahl .... (steht auf jedem Bankauszug)

Vorname .... Nachname ....

Str./Nr. .... PLZ / Ort ....

Verkaufsbedingungen: Versand nur gegen Vorkasse oder Bankabbuchung. Keine Nachnahme!

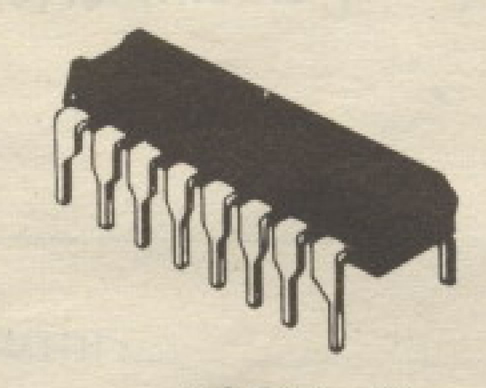
Unterschrift ....

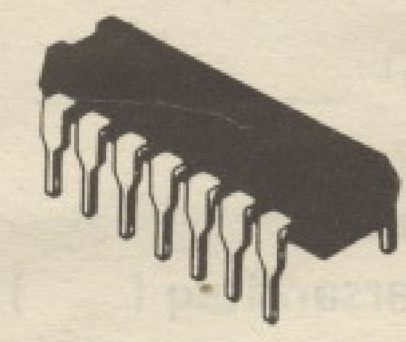
Bitte ausschneiden und einsenden an
TI-REVUE
Special-Service 8/86
Postfach 1107
8044 Unterschleißheim



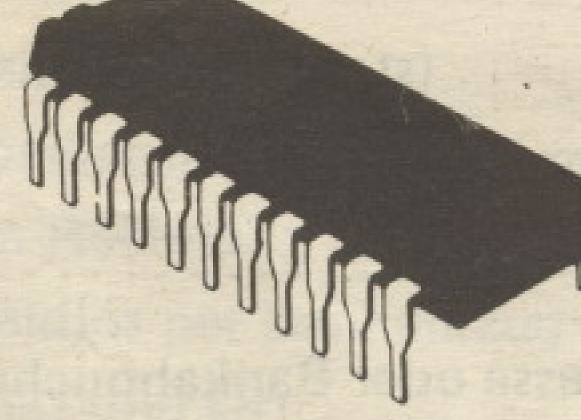
Gehäuse und thermische Angaben

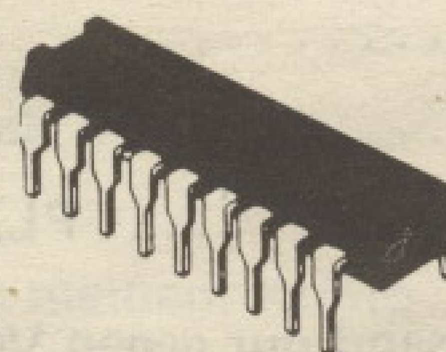
PLASTIK

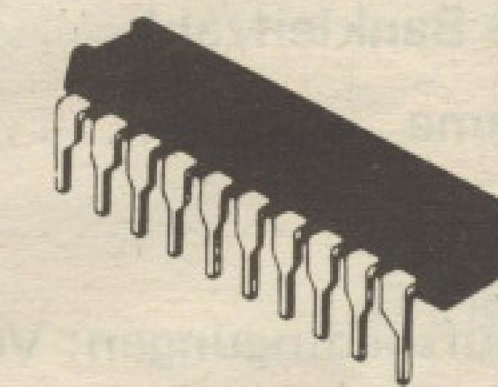

16 PIN – N

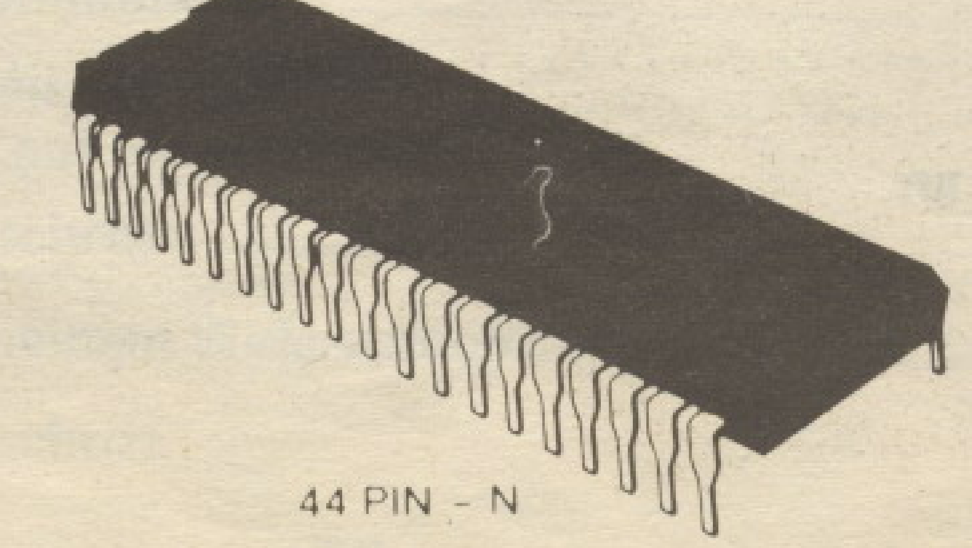

14 PIN – N

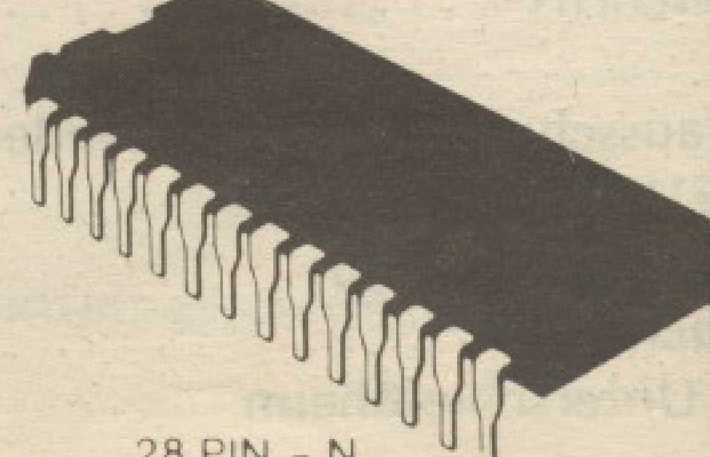

8 PIN – N


22 PIN – N

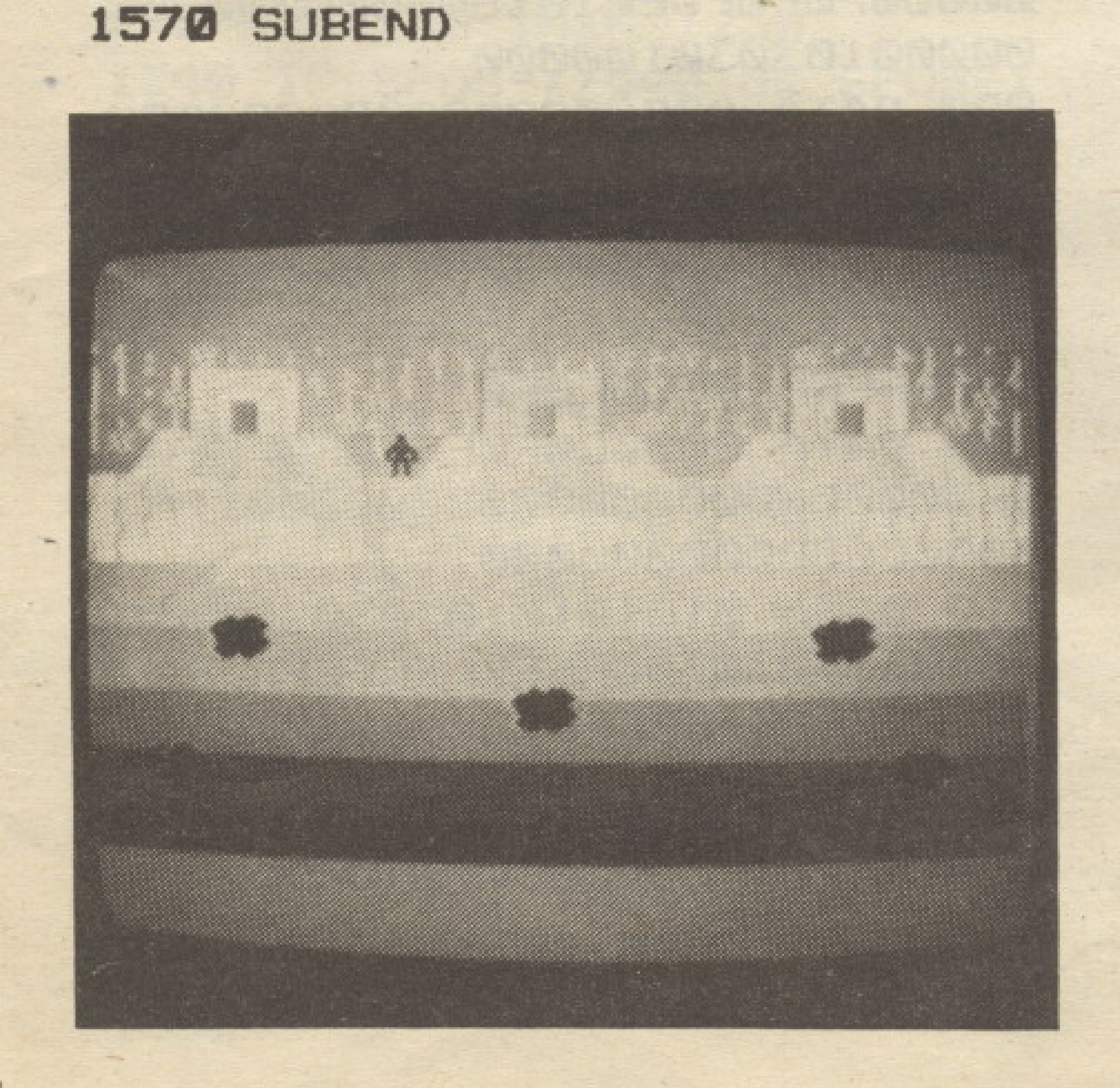

18 PIN – N


20 PIN – N


44 PIN – N


28 PIN – N

```
REEN(16):: CALL SOUND(-99,-5
,A)
1420 CALL SCREEN(7):: CALL S
CREEN(2):: NEXT A :: SUBEND
1430 SUB Y(A,B,C$):: DISPLAY
 AT(A,B)SIZE(LEN(C$)):C$ ::
SUBEND
1440 SUB X(X):: X$=STR$(X)::
 DISPLAY AT(12,13)SIZE(LEN(X
$)):X$ :: SUBEND
1450 SUB W
1460 DATA 196,294,196,294,19
6,294,196,294,233,349,233,34
9,233,349
1470 DATA 233,349,175,262,17
5,262,175,262,175,262,196,29
4,196,294,196,294,196,294
1480 DATA 196,294,196,294,19
6,330,196,294,196,349,196,29
4,196,330,196,294
1490 DATA 196,294,196,294,19
6,330,196,294,196,349,196,29
4,196,330,196,294
1500 DATA 233,349,233,349,23
3,392,233,349,233,415,233,34
9,233,392,233,349
1510 DATA 233,349,233,349,23
3,392,233,349,233,415,233,34
9,233,392,233,349
1520 DATA 175,262,175,262,17
5,294,175,262,175,311,175,26
2,175,294,175,262
1530 DATA 175,262,175,262,17
5,294,175,262,175,311,175,26
2,175,294,175,262
1540 DATA 196,294,196,294,19
6,330,196,294,196,349,196,29
4,196,330,196,294
1550 DATA 196,294,196,294,19
6,330,196,294,196,349,196,29
4,196,330,196,294
1560 RESTORE 1460 :: FOR A=1
 TO 80 :: READ B,C :: CALL S
OUND(225,B,2,C,1):: NEXT A
1570 SUBEND
```



LISTINGS

```
10 ! **********************
11 ! *                    *
12 ! *    VERSCHIEBEN     *
13 ! *                    *
14 ! *   Copyright by     *
15 ! *                    *
16 ! *   Frank Albert     *
17 ! *                    *
19 ! * Benoetigte Geraete *
20 ! *  TI99/4A  Konsole  *
21 ! *     Ext. Basic     *
22 ! *    Joystick 2      *
23 ! *                    *
26 ! *  Speicherbelegung  *
27 ! *     6015 Bytes     *
28 ! *                    *
29 ! **********************
100 CALL CLEAR :: RANDOMIZE
110 OPTION BASE 1 :: DIM SOL
L(4,4),IST(4,4)
120 FOR I=1 TO 4 :: FOR J=1
TO 4 :: READ SOLL(I,J):: IST
(I,J)=SOLL(I,J):: NEXT J ::
NEXT I
130 DATA 1,2,3,4,5,6,7,8,9,1
0,11,12,0,13,14,15
140 CALL DEFINIERE
150 !
160 ! SPIELANLEITUNG
170 !
180 DISPLAY AT(1,9):"O>KL<AB
>;>G" :: DISPLAY AT(2,8):"--
------------"
190 DISPLAY AT(6,1):"O>KLN<A
> =:L NKLIKN>G@EB<A> ;BE= =N
K<A O>KL<AB>;>G =>K  >BGS>EG
>G L>@F>GM>,:NL =>G>G=:L ;BE
= ;>LM>AM,PB>=>K A>K-SNLM>EE
>G!"
200 DISPLAY AT(13,1):"@>LM>N
>KM PBK= FBM CHRLMB<D GNFF>K
 2!"
210 DISPLAY AT(17,1):"=:L LI
B>E BLM SN >G=>,P>GG..=N F>A
K :EL =HII>EM LH OB>E>SN>@>
;K:N<ALM,:EL B<A SNF..FBL<A>
G ;>GH>MB@M A:;>!"
220 DISPLAY AT(24,4):"*?>N>K
DGHI? =KN><D>G*"
230 CALL MUSIK
240 CALL KEY(2,K,S):: IF S=0
 THEN 240
250 CALL MUSIK
260 !
270 ! BILDAUFBAU
280 !
290 CALL COLOR(2,16,13)
300 CALL CLEAR
310 ZEILE=4 :: SPALTE=1
320 CALL HCHAR(2,4,46,18)::
CALL VCHAR(3,21,46,17):: CAL
L HCHAR(19,4,46,17):: CALL V
CHAR(3,4,46,16)
```

```
330 FOR I=3 TO 18 :: CALL HC
HAR(I,5,47,16):: NEXT I
340 DISPLAY AT(2,21):"KNG=>"
 :: DISPLAY AT(7,23):"L<HK>"
 :: DISPLAY AT(12,21):"ABL<H
K>"
350 DISPLAY AT(4,25):"0" ::
DISPLAY AT(9,27):"0" :: DISP
LAY AT(14,27):"0"
360 CALL AUFBAU(IST(,))
370 DISPLAY AT(23,2):"*?>N>K
DGHI? =KN><D>G*"
380 CALL KEY(2,K,S):: IF S=0
 THEN 380
390 !
400 ! SPIELBEGINN
410 !
420 DISPLAY AT(21,2):"" :: D
ISPLAY AT(23,2):""
430 RUNDE=RUNDE+1 :: DISPLAY
 AT(4,24):USING "##":RUNDE
440 ZUEGE=0
450 CALL MISCHE(IST(,),ZEILE
,SPALTE,RUNDE):: CALL AUFBAU
(IST(,))
460 CALL VERSCHIEBE(IST(,),Z
EILE,SPALTE)
470 ZUEGE=ZUEGE+1
480 CALL VERGLEICHE(SOLL(,),
IST(,),IDENTISCH)
490 IF IDENTISCH THEN 650 EL
SE IF ZUEGE<2*RUNDE THEN 460
500 !
510 ! SPIELENDE
520 !
530 DISPLAY AT(21,2):"* LIB>
E>G=> *"
540 FOR I=200 TO 110 STEP -5
 :: CALL SOUND(-300,I,5):: N
EXT I
550 DISPLAY AT(23,2):"* ?>N>
KDGHI? =KN><D>G *"
560 IF SCORE>HISCORE THEN HI
SCORE=SCORE :: DISPLAY AT(14
,23):USING "#####":HISCORE :
: CALL MUSIK
570 CALL KEY(2,K,S):: IF S=0
 THEN 570
580 FOR I=1 TO 4 :: FOR J=1
TO 4 :: IST(I,J)=SOLL(I,J)::
 NEXT J :: NEXT I
590 ZEILE=4 :: SPALTE=1
600 DISPLAY AT(21,2):"" :: D
ISPLAY AT(23,2):""
610 RUNDE,SCORE=0
620 CALL AUFBAU(IST(,))
630 DISPLAY AT(9,23):"....0"
640 GOTO 430
650 !
660 ! BILD WIEDER KOMPLETT
670 !
680 CALL SOUND(100,110,5,220
,5):: CALL SOUND(150,147,0,2
94,0)
690 DISPLAY AT(21,2):"?>KMB@
!"
700 DISPLAY AT(23,2):"* ?>N>
KDGHI? =KN><D>G *"
710 SCORE=SCORE+INT(RUNDE*20
/ZUEGE)*RUNDE :: DISPLAY AT(
9,23):USING "#####":SCORE
720 GOTO 380
730 !
740 ! PROZEDUREN
750 !
760 SUB DEFINIERE
770 DATA 000000000000000000
00000000000003F3F3F3F3F7FF
FFEFCF9F9E9E9FCFE
780 DATA FFFFFFFFFFFFFFF0F0
300C0C09060A0FFFFFFFFFFFFFF
FFFFF0F0F0F1FAF
790 DATA FCFCFCFCFCFFFFFFFFF
F8000000000000000000000F0F8F
CF8C0000000003777
800 DATA 000000000000000000
000000000000070707070707070
7030101010101010101
810 DATA 000000000000000000
000000000007F7F7F7F7F3F3F3
F3F3F3F3733331919
820 DATA FEFDFFFFFFFFFFFFFFF
FFFFFFFFFFFF7FFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
830 DATA 000100010100000000
000000000000090F0E0F0F0FEFFF
FFEFEFC7C3C3C1C0C
840 DATA 000000000000000000
000000000000101010303030303030
303030707070F0F0F
850 DATA 000000000000000000
000000000000D07070303030303030
3030303010101010000
860 DATA FFFFFFFFFFFFFFFFFF
FFFFEFFFFFFFCFC0F0FEFEFEFEF
CFCF8E000FFFFFFFF
870 DATA 000000008080800C0E08
00000F0F0F0E018180800000000
00000103030100000
880 DATA 000000000000000000
000000001010011F3F3F7FFFFFFFF
FFFFFFFFFFFFFFFF
890 DATA FFFF7F7F7F7F3F3F3F3
F3F3F3F3F3F3FF8FCFFFFFFFFFFF
FFFFFFFFF9EC0F0F8
900 DATA 0000C0F0F0F0F0F0F
0E000000000000000000000000000
000000000000000000
910 DATA 01010101030303030
303030303030303FFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
920 FOR I=65 TO 90 :: CALL C
HARPAT(I,CHAR$):: CALL CHAR(
I-7,CHAR$):: NEXT I
930 CALL CHAR(46,"FFFFFFFFFF
FFFFFF",47,"")
```

LISTINGS

```
940 RESTORE 770 :: FOR I=84
TO 140 STEP 4 :: READ CODE$
:: CALL CHAR(I,CODE$):: NEXT
 I
950 CALL SCREEN(2):: CALL MA
GNIFY(4):: CALL COLOR(1,6,1,
2,6,1,3,6,1,4,6,1,5,6,1,6,6,
1,7,6,1)
960 FOR I=1 TO 15 :: CALL SP
RITE(#I,I*4+80,12,193,1):: N
EXT I
970 SUBEND
980 !
990 ! *********************
1000 !
1010 SUB AUFBAU(FELD(,))
1020 FOR ZEILE=1 TO 4
1030 FOR SPALTE=1 TO 4
1040 IF FELD(ZEILE,SPALTE)TH
EN CALL LOCATE(#FELD(ZEILE,S
PALTE),ZEILE*32-15,SPALTE*32
+1)
1050 NEXT SPALTE
1060 NEXT ZEILE
1070 SUBEND
1080 !
1090 ! *********************
1100 !
1110 SUB VERGLEICHE(FELD1(,)
,FELD2(,),IDENTISCH)
1120 IDENTISCH=-1
1130 FOR I=1 TO 4
1140 FOR J=1 TO 4
1150 IF FELD1(I,J)<>FELD2(I,
J)THEN IDENTISCH=0
1160 NEXT J
1170 NEXT I
1180 SUBEND
1190 !
1200 ! *********************
1210 !
1220 SUB VERSCHIEBE(FELD(,),
ZPOS,SPOS)
1230 CALL JOYST(2,X,Y):: IF
X=0 AND Y=0 THEN 1230
1240 IF SPOS-X/4=0 OR SPOS-X
/4=5 OR Y/4+ZPOS=0 OR Y/4+ZP
OS=5 THEN 1230
1250 CALL SOUND(650,-5,15)
1260 SPRITE=FELD(ZPOS+Y/4,SP
OS-X/4):: CALL POSITION(#SPR
ITE,SZEILE,SSPALTE)
1270 FOR I=1 TO 16 :: CALL L
OCATE(#SPRITE,SZEILE-I*Y/2,S
SPALTE+I*X/2):: NEXT I
1280 CALL SWAP(FELD(ZPOS+Y/4
,SPOS-X/4),FELD(ZPOS,SPOS))
1290 ZPOS=ZPOS+Y/4 :: SPOS=S
POS-X/4
1300 SUBEND
1310 !
1320 ! *********************
1330 !
```

LISTINGS

```
1340 SUB SWAP(X,Y)
1350 HILF=X :: X=Y :: Y=HILF
1360 SUBEND
1370 !
1380 ! *********************
1390 !
1400 SUB MISCHE(FELD(,),ZPOS
,SPOS,N)
1410 FOR I=1 TO N
1420 X0=INT(RND*4)+1 :: IF A
BS(X0-X)=2 THEN 1420
1430 ON X0 GOTO 1440,1450,14
60,1470
1440 IF ZPOS=1 THEN 1420 ELS
E CALL SWAP(FELD(ZPOS,SPOS),
FELD(ZPOS-1,SPOS)):: ZPOS=ZP
OS-1 :: GOTO 1480
1450 IF SPOS=1 THEN 1420 ELS
E CALL SWAP(FELD(ZPOS,SPOS),
FELD(ZPOS,SPOS-1)):: SPOS=SP
OS-1 :: GOTO 1480
1460 IF ZPOS=4 THEN 1420 ELS
E CALL SWAP(FELD(ZPOS,SPOS),
FELD(ZPOS+1,SPOS)):: ZPOS=ZP
OS+1 :: GOTO 1480
1470 IF SPOS=4 THEN 1420 ELS
E CALL SWAP(FELD(ZPOS,SPOS),
FELD(ZPOS,SPOS+1)):: SPOS=SP
OS+1
1480 X=X0
1490 NEXT I
1500 SUBEND
1510 !
1520 ! *********************
1530 !
1540 SUB MUSIK
1550 CALL SOUND(100,131,0,26
2,0)
1560 CALL SOUND(100,147,0,29
4,0)
1570 CALL SOUND(100,165,0,33
0,0)
1580 CALL SOUND(100,262,0,52
3,0)
1590 FOR I=1 TO 45 :: NEXT I
1600 CALL SOUND(100,220,0,44
0,0)
1610 FOR I=1 TO 45 :: NEXT I
1620 CALL SOUND(100,247,0,49
4,0)
1630 FOR I=1 TO 45 :: NEXT I
1640 CALL SOUND(100,196,0,39
2,0)
1650 FOR I=1 TO 45 :: NEXT I
1660 CALL SOUND(100,175,0,34
9,0)
1670 FOR I=1 TO 20 :: NEXT I
1680 CALL SOUND(100,196,0,39
2,0)
1690 FOR I=1 TO 30 :: NEXT I
1700 CALL SOUND(100,165,0,33
0,0)
1710 FOR I=1 TO 30 :: NEXT I
```

```
1720 CALL SOUND(250,131,0,26
2,0)
1730 SUBEND
1740 !
1750 ! *********************
1760 !
1770 END
```

ANDERE DRUCKER VON TI LOGO AUS BENUTZEN

Ist es möglich, mit dem TI LOGO Modul auch andere Druckerschnittstellen anzusprechen außer dem TI Thermoprinter?
Peter Danzeisen, Hamburg

Selbstverständlich ist es möglich, aus TI LOGO auch andere Drucker anzusprechen. Sie müssen nur immer dort, wo bisher 'TP' steht, den Namen Ihrer Schnittstelle eingeben. Verwenden Sie z.B. die PIO-Schnittstelle, so geben Sie einfach 'PIO' ein.

MIC: NICHT NUR EIN DISK-INFO KATALOG!

MIC liefert neben den Informationen, die ein übliches Disk-Katalog-Programm auch liefert, nämlich
Diskettenname
Anzahl der freien Sektoren (F=xxx)
Anzahl der belegten Sektoren (U=xxx)
Filenamen
Filelängen in Sektoren
Fileformate
Satzlängen,
mit Hilfe des Disk-Managers geschützter (P) oder ungeschützter (U) File auch zusätzliche File-Informationen:
– ein erläuternder File-Info-Text zu jedem File, falls ein solcher zuvor – ebenfalls mittels MIC – auf die Diskette geschrieben wurde (max. 32 Zeichen)
– ob Programm-Format-Basicprogramm (PGM/BU falls nicht geschützt, PGM/BP falls geschützt durch 'protected' in der Basic-SAVE-Anweisung)
– oder ob Programm-Format-Maschinenprogramm (PGM/MA), z.B. erstellt mit dem SAVE-Programm des Editor/Assembler-Paketes und startbar mit Run Program File
– welche Sektoren der File belegt (aaa-bbb ccc-ddd... in hexadezimalen Zahlen, weil die bekannten Disk-Fixer solche als Eingabe benötigen)
– bei PGM/MA-Files die ersten fünf Wörter in hexadezimaler Darstellung, denn diese enthalten Informationen über Ladeadresse und Länge, Einsprungadresse und eventuelle Anschlußfiles. Sie lassen weiterhin erkennen, ob der File der Run-Program-File-Konvention entspricht, ob es sich um einen Modul-Inhalt handelt.

Zusätzliche Disk-Informationen:
– ob geschützte (P) oder ungeschützte (U) Diskette 'proprietary protection'
– ob einseitig (SI=1) oder doppelseitig (SI=2) initialisierte Diskette
– ob einfache (D=1) oder doppelte (D=2) Schreibdichte
– Anzahl der Spuren/Seite (TR/SI=xx)
– Anzahl der Sektoren/Spur (SE/TR=xx)
– Anzahl der Files auf der Diskette
– Summe der von Files 'offiziell' belegten Sektoren

MIC-GEHEIMNISSE

MIC ist ein Extended-Basic-Programm mit darin 'versteckten' Maschinensprache-Unterprogrammen (Lesen und Schreiben im VDP-RAM und Lesen und Schreiben von Diskettensektoren). Die Unterprogramme werden also weder durch CALL-LOAD-Aufrufe noch durch Poken aus DATA-Zeilen geladen. Sie sind auf ihren Plätzen, sobald das Basicprogramm geladen ist. Damit die Unterprogramme in einem von der CPU direkt erreichbaren Speicherbereich liegen, muß eine Speichererweiterung angeschlossen sein.
Derartige Programme findet man in letzter Zeit häufiger. Deshalb für Interessierte hier ein paar Hinweise darauf, wie man so etwas macht. Im Ur-Basicprogramm werden einige Zeilen vorgesehen, die nur als Platzhalter dienen, am einfachsten Kommentarzeilen. Der so reservierte Platz im Speicher muß ausreichend sein, die Unterprogramme aufzunehmen. In MIC waren die Zeilen 180, 190 und 200 ursprünglich lange Kommentarzeilen. Das vorbereitete Basicprogramm wird geladen, und mit Hilfe von DEBUG wird nachgesehen, ab welcher Adresse diese Platzhalter im Speicher stehen. Dann kann mittels einer entsprechenden AORG-Anweisung im Assemblerprogramm dafür gesorgt werden, daß den Unterprogrammen die richtigen Plätze zugewiesen werden. Mit DEBUG wird weiterhin

geschaut, auf welchen Speicherplätzen in der Zeilenliste die Adressen der Platzhalter-Zeilen stehen und welche Adresse eine Ersatzzeile hat, die formal statt der überschriebenen Platzhalter-Zeilen dienen kann. Als ein solcher Ersatz dient in MIC die kurze Kommentarzeile 160. Die Adressen nach den Platzhalterzeilennummer (in MIC 180, 190 und 200) werden später mit der Adresse der Ersatzzeile (in MIC 160) überschrieben.
Es muß dafür gesorgt werden, daß die Unterprogramme bei CALL-LINK-Aufrufen auch gefunden werden. Deshalb wird vom Basicprogramm aus die REF/DEF-Tafel am Ende des LOW MEMORY durch Poken erstellt. Die Anfangsadresse der REF/DEF-Tafel muß nach Adresse 8196 geschrieben werden. In MIC geschieht das alles in Zeile 280 unter Verwendung der DATA-Zeilen 220 und 230.
Beim Entstehungsprozeß sollte nach Änderungen im Ur-Basicprogramm dieses wieder in eine im Speicher wohlgeordnete Form gebracht werden, z.B. mit der Anweisungsfolge
SAVE DSK1.UR_MIC/M,MERGE
NEW
MERGE DSK1.UR_MIC/M
SAVE DSK1.UR_MIC

Nach den Vorbereitungen – die komplizierter klingen als sie sind – entsteht das lauffähige, fertige Programm (hier MIC als Beispiel) durch die Anweisungen:

OLD DSK1.UR_MIC
(noch ohne Unterprogramme)
CALL INIT
(damit CALL LOAD funktioniert)
CALL LOAD
("DSK1.MIC_UP")
(das assemblierte Unterprogrammpaket)
CALL LOAD
(–4718,254,158)
(Zeile 200 wie Zeile 160)
CALL LOAD
(–4714,254,158)
(Zeile 190 wie Zeile 160)
CALL LOAD
(–4710,254,158)
(Zeile 180 wie Zeile 160)
SAVE DSK1.MIC
(Abspeichern des fertigen Programms)

Eine Liste des Assembler-Unterprogrammpaketes wird ohne Kommentare beigefügt. Die darin enthaltenen Routinen PEEKV und POKEV sind in ähnlicher Form schon verschiedentlich veröffentlicht worden.
Die Routinen XRSECT und XWSECT stammen vom Autor. Sie funktionieren unabhängig vom Typ des verwendeten Disk-Controllers.
Hier noch die Erklärung, wie und wo die 32 Zeichen File-Info-Text gespeichert werden, die mit dem Katalog bei jedem File ausgedruckt werden können. Zu jedem File gibt es auf der Diskette einen Directory-Sektor, der den Filenamen sowie Informationen über Art und Länge enthält. Außerdem steht dort, welche Sektoren der File belegt. Für letzteres ist sehr viel Platz vorhanden, so viel sogar, daß er selbst bei einem außerordentlich stark segmentierten File (der Inhalt ist über viele Bereiche der Diskette verstreut) wohl nie ausgenutzt wird. Deshalb wird das hinterste Ende dieses Platzes, die letzten 32 Bytes jedes Directory-Sektors nämlich, für den zusätzlichen Info-Text verwendet. Diese File-Information kostet also keinen Extraplatz auf der Diskette.

DIE BENUTZUNG VON MIC

Nach dem Programmstart (mit RUN "DSKx.MIC") erscheint die Auswahlliste

MULTI-INFO-CATALOG
1. DISPLAY/PRINT CATALOG
2. READ/WRITE FILE INFO
3. READY
YOUR CHOICE: 1

Wird 1 (DISPLAY/PRINT CATALOG) gewählt, so erscheint am Schirm:

CATALOG
FROM DRIVE #
TO PRT-DEV.

Als Laufwerksnummer kann 1 bis 4 oder –1 bis –4 oder 0 eingegeben werden. Bei positiver Nummer werden alle Informationen ausgegeben, bei negativer Nummer nur ein Teil. Die Null führt zurück zur ersten Auswahlliste.
Als Print-Device ist jede gültige Beschreibung zulässig, z.B. PIO oder PRINT oder RS232. DA=8.BA=4800 aber auch DSK3.AUSGABE (wozu auch immer Letzteres gut sein mag). Wird ohne Eingabe nur ENTER gedrückt, so wird anfangs PIO angenommen und später das zuletzt eingegebene Print-Device. Bei Eingabe von S erfolgt die Ausgabe am Bildschirm. Mit einer beliebigen Taste kann die Ausgabe angehalten und wieder fortgesetzt werden.
Am Ende des Kataloges wird die Anzahl yy der Files (=Anzahl der Directory-Sektoren) und die Anzahl zz der von diesen Files belegten Sektoren ausgegeben &U=yy+zz). Durch Vergleich dieser Summe mit der Anzahl der laut Sektor 0 belegten Sektoren (U=xxx) in der Kopfzeile des Kataloges kann auf eventuelle verborgene Sektoren geschlossen werden, auf die raffiniert geschützte Programme direkt zugreifen. Am Katalogende kehrt das Programm mit beliebiger Taste zur ersten Auswahlliste zurück.

Wird 2 (READ/WRITE FILE INFO) angewählt, so erscheint am Schirm:

READ/WRITE FILE INFO
DRIVE #1
FILE: EACH

Als Laufwerksnummer kann 0 bis 4 eingegeben werden. Die Null führt auch hier zur ersten Auswahlliste zurück.
EACH kann mit einem Filenamen überschrieben werden. Dann wird der entsprechende Directory-Sektor auf der Diskette gesucht und ein eventuell schon existierender Informationstext gelesen und auf den Schirm geschrieben. Dieser kann dann überschrieben bzw. erstmals geschrieben werden. ENTER bewirkt das Rückschreiben auf die Diskette. Außerdem sind die Tasten ERASE (Fctn 3), LINKS- und RECHTSPFEIL, BEGIN (Fctn 5) und BACK (Fctn 9) sowie QUIT (Fctn =) wirksam. ERASE löscht den ganzen Text, die Pfeile sind selbsterklärend, BEGIN und BACK führen zurück in die letzte Auswahlliste, QUIT bricht das Programm ab.
Wird EACH gewählt, so werden alle File-Informationstexte der Diskette nacheinander ausgelesen und können überschrieben bzw. erstmals geschrieben werden. Hier kann die Verwendung der Taste REDO sinnvoll sein, die bewirkt, daß der beim letzten File benutzte Text ohne erneute Eingabe wiederbenutzt wird. Wird ca. zwölf Sekunden lang keine Taste gedrückt, so geht das Programm zum nächsten File weiter.
Das Beschreiben einer Diskette mit den zusätzlichen File-Info-Texten ist erst dann zweckmäßig, wenn sie zu einer Archiv-Diskette ausgegoren ist. Jede Änderung eines Files und jedes Kopieren – wobei ja das Betriebssystem den Directory-Sektore auf den neuesten Stand bringt – löscht den Informationstext. Nur solche Kopierprogramme, die sektorweise oder spurweise kopieren (z.B. QUICKCOPY, SECTOR-COPY, TURBO u.a.), lassen diese Texte bestehen.

Dr. W. Jüngst

```
100 ! PROGRAMMNAME: MIC (MUL
TI-INFO-CATALOG)
110 ! BENOETIGT 32K-ERWEITER
UNG
120 ! LIEFERT: SEKTOR-Ø-INFO
RMATION, BU/BP/MA BEI PROGRA
M,
130 ! 5 ANFANGSWOERTER BEI M
A-PGM, SEKTORENBELEGUNG JEDE
S FILES SOWIE
140 ! FILE-INFORMATION (LETZ
TE 32 BYTES DES DIRECTORY-SE
KTORS).
150 ! ERMOEGLICHT LESEN UND
SCHREIBEN/MODIFIZIEREN DIESE
R FILE-INFORMATION.
160 !
170 ! W.JUENGST (1/86)
180 !
                    *
190 !
                    *
200 !
                          *
210 DIM A(10),TYP$(5)
220 DATA 88,82,83,69,67,84,2
53,130,88,87,83,69,67,84,253
,120
230 DATA 80,69,69,75,86,32,2
53,236,80,79,75,69,86,32,254
,44
240 DRO$="PIO" :: GOTO 270 :
: CALL INIT :: CALL LOAD ::
CALL KEY :: CALL LINK :: CAL
L PEEK :: CALL CLEAR :: CALL
 HCHAR
250 A$,B$,C$,BA$,C$,DN$,DR$,
H$,N$,P$,SI$=Z$ :: B,C,DN,FN
,FT,I,J,K,L,M,N,NO,P,S,S1,ST
,X,XO,XT,Y=Z
260 !@P-
270 Z$="Ø123456789ABCDEF" ::
 TYP$(1)="DIS/FIX " :: TYP$(
2)="DIS/VAR " :: TYP$(3)="IN
T/FIX " :: TYP$(4)="INT/VAR
" :: TYP$(5)="PROG"
280 CALL INIT :: CALL LOAD(8
196,63,224):: FOR C=16352 TO
 16383 :: READ B :: CALL LOA
D(C,B):: NEXT C
290 DISPLAY AT(6,2)ERASE ALL
:"***MULTI-INFO-CATALOG***"
:: DISPLAY AT(12,2):"1. DISP
LAY/PRINT CATALOG"
300 DISPLAY AT(14,2):"2. REA
D/WRITE FILE INFO" :: DISPLA
Y AT(16,2):"3. READY" :: DIS
PLAY AT(21,8):"YOUR CHOICE:
1"
310 ACCEPT AT(21,21)VALIDATE
("123")SIZE(-1):J :: IF J=3
THEN STOP ELSE IF J=2 THEN 7
80 ELSE CALL CLEAR
320 NO=Ø :: PRINT : :"CATALO
G" :: INPUT "FROM DRIVE #":D
N :: IF DN<Ø THEN DN=-DN ::
NO=-1
330 IF DN=Ø OR DN>4 THEN 290
340 INPUT "TO PRT-DEV. ":DR$
 :: DN$=STR$(DN):: FN=2 :: F
T=Ø :: XT=Ø
350 IF DR$="" THEN DR$=DRO$
:: DISPLAY AT(23,13):DR$
360 DRO$=DR$ :: IF DR$="S" T
HEN FN=Ø :: GOTO 370 ELSE OP
EN #2:DR$
370 OPEN #1:"DSK"&DN$&".",IN
PUT ,RELATIVE,INTERNAL :: IN
PUT #1:N$,J,J,K
380 PRINT #FN: :"DSK=";N$;TA
B(16);"F=";STR$(K);TAB(23);"
U=";STR$(J-K)
390 CALL LINK("PEEKV",15611,
I,X,X,X,J,K,L,M):: IF J=32 T
HEN J=85
400 PRINT #FN:"SI="&STR$(L)&
" TR/SI="&STR$(K)&" SE/TR="&
STR$(I)&" D="&STR$(M);TAB(28
);CHR$(J):: PRINT #FN:RPT$("
-",28)
410 CALL KEY(Ø,X,ST):: IF ST
<>1 THEN 430
420 CALL KEY(Ø,X,ST):: IF ST
<>1 THEN 420
430 BA$="/MA " :: INPUT #1:N
$,I,J,K :: SI$="" :: IF LEN(
N$)=Ø THEN 730
440 PRINT #FN:N$;TAB(12);STR
$(J);TAB(16);TYP$(ABS(I));::
 IF NO THEN 510
450 XO=-1 :: FOR ST=15627 TO
 15657 STEP 3 :: CALL LINK("
PEEKV",ST,L,M,N)
460 S=L+256*(M-16*INT(M/16))
 :: IF S=Ø THEN 500
470 IF ST=15627 THEN S1=S
480 Z=S :: GOSUB 760 :: SI$=
SI$&H$&"-" :: X=INT(M/16)+16
*N :: Z=S+X-XO-1
490 GOSUB 760 :: SI$=SI$&H$&
" " :: XO=X :: NEXT ST :: SI
$=SI$&"*"
500 FT=FT+1 :: XT=XT+XO+1
510 CALL LINK("PEEKV",15823,
A(Ø),A(1)):: IF A(Ø)=Ø THEN
C$="" :: GOTO 550 ELSE C$=CH
R$(A(Ø))&CHR$(A(1))
520 FOR X=Ø TO 2
530 CALL LINK("PEEKV",15825+
10*X,A(Ø),A(1),A(2),A(3),A(4
```

LISTINGS

```
),A(5),A(6),A(7),A(8),A(9))
540 FOR S=0 TO 9 :: C$=C$&CH
R$(A(S)):: NEXT S :: NEXT X
550 B$="" :: IF ABS(I)=5 THE
N IF NO THEN 660 ELSE 570
560 PRINT #FN:STR$(K);:: GOT
O 660
570 CALL LINK("XRSECT",DN,S1
)
580 CALL LINK("PEEKV",15599,
A(1),A(2),A(3),A(4),A(5),A(6
),A(7),A(8),A(9),A(10))
590 L=A(2)+256*A(1):: IF L=0
 THEN 630
600 M=A(4)+256*A(3):: N=A(6)
+256*A(5):: IF M>32767 OR N>
32767 OR M=N THEN 630
610 P$="U " :: IF L>32767 TH
EN L=65536-L :: P$="P "
620 IF (N XOR M)=L THEN BA$=
"/B"&P$
630 PRINT #FN:BA$;:: IF BA$<
>"/MA " THEN 660
640 FOR S=1 TO 9 STEP 2 :: F
OR L=0 TO 1 :: Z=A(S+L):: M=
INT(Z/16):: N=Z-16*M
650 B$=B$&SEG$(Z$,M+1,1)&SEG
$(Z$,N+1,1):: NEXT L :: B$=B
$&" " :: NEXT S
660 P$="U" :: IF I<0 THEN P$
="P"
670 PRINT #FN:TAB(28);P$;::
L=-1
680 IF C$<>"" THEN PRINT #FN
:TAB(FN*17);C$ :: L=0
690 IF SI$<>"" THEN PRINT #F
N:TAB(FN*17);SI$ :: L=0
700 IF B$<>"" THEN PRINT #FN
:TAB(FN*17);B$ :: L=0
710 IF L THEN PRINT #FN
720 IF FN THEN 410 ELSE PRIN
T :: GOTO 410
730 IF NOT NO THEN PRINT #FN
:TAB(FN*17);"U="&STR$(FT)&"+
"&STR$(XT)
740 CALL KEY(0,X,ST):: IF ST
<>1 THEN 740
750 CLOSE #1 :: IF FN>0 THEN
 CLOSE #2 :: GOTO 290 ELSE 2
90
760 H$="" :: P=256 :: FOR J=
1 TO 3 :: Y=INT(Z/P):: H$=H$
&SEG$(Z$,Y+1,1):: IF P=1 THE
N RETURN
770 Z=Z-Y*P :: P=P/16 :: NEX
T J
780 CALL LOAD(-31806,16):: D
ISPLAY AT(2,1)ERASE ALL:"REA
D/WRITE FILE INFO"
790 DISPLAY AT(4,1):"DRIVE #
1" :: DISPLAY AT(6,1):"FILE:
 EACH"
800 ACCEPT AT(4,8)VALIDATE("
01234")SIZE(-1):DN :: IF DN=
0 THEN 290
810 ACCEPT AT(6,7)SIZE(-10):
A$ :: L=LEN(A$)
820 B$="" :: NO=0 :: IF A$="
EACH" THEN NO=-1
830 FOR FN=15599 TO 15853 ST
EP 2 :: S1=1 :: CALL LINK("X
RSECT",DN,S1)
840 CALL LINK("PEEKV",FN,B,C
):: S1=256*B+C :: IF S1=0 TH
EN 790
850 CALL LINK("XRSECT",DN,S1
):: IF NO THEN 890
860 FOR N=1 TO L :: CALL LIN
K("PEEKV",15598+N,J)
870 IF J<>ASC(SEG$(A$,N,1))T
HEN 1120
880 NEXT N :: GOTO 910
890 A$="" :: FOR N=1 TO 10 :
: CALL LINK("PEEKV",15598+N,
J):: IF J=32 THEN 910
900 A$=A$&CHR$(J):: NEXT N
910 DISPLAY AT(6,7):A$
920 DISPLAY AT(10,1):"INFO-T
EXT IN DIRECTORY:"
930 FOR I=1 TO 32 :: CALL LI
NK("PEEKV",15822+I,J)
940 CALL LINK("POKEV",351+I,
J+96):: NEXT I :: M=0
950 I=0
960 IF I<32 THEN I=I+1
970 CALL HCHAR(13,1,32,32)::
 CALL HCHAR(13,I,45):: P=0
980 CALL KEY(0,X,ST):: P=P+1
 :: IF ST=1 THEN 990 ELSE IF
 P<300 THEN 980 ELSE 1060
990 IF X=13 THEN 1060 ELSE I
F X=7 THEN 950 ELSE IF X=14
OR X=15 THEN 780
1000 IF X<>6 THEN 1020
1010 CALL HCHAR(12,1,32,32):
: FOR I=1 TO LEN(B$):: CALL
HCHAR(12,I,ASC(SEG$(B$,I,1))
):: NEXT I :: M=-1 :: GOTO 9
50
1020 IF X=5 THEN STOP ELSE I
F X<>8 THEN 1040
1030 IF I>1 THEN I=I-1 :: GO
TO 970 ELSE GOTO 980
1040 IF X=9 THEN 960
1050 CALL HCHAR(12,I,X):: M=
-1 :: GOTO 960
1060 IF NOT M THEN 1100
1070 DISPLAY AT(16,1):"REWRI
TING TEXT" :: B$="" :: FOR I
=1 TO 32
1080 CALL LINK("PEEKV",351+I
,J):: J=J-96 :: CALL LINK("P
OKEV",15822+I,J):: B$=B$&CHR
$(J):: NEXT I
1090 CALL LINK("XWSECT",DN,S
```

```
1)
1100 IF NOT NO THEN 780
1110 CALL HCHAR(16,1,32,16):
: CALL HCHAR(12,1,32,64)
1120 NEXT FN
```

ASSEMBLER TEIL

LISTINGS

```
* MIC_UP(S) READ/WRITE DSK-SECTOR
*           AND PEEKV/POKEV FROM XB
* FOR MIC (MULTI-INFO-CATALOG)
* CALL LINK("XRSECT",DRIVE#,SECTOR#)
* CALL LINK("XWSECT",DRIVE#,SECTOR#)
* SECTOR-CONTENTS IN VDP-LOCATIONS
* STARTING FROM >3CEF
* CALL LINK("PEEKV",ADDR,X1,X2,...)
* CALL LINK("POKEV",ADDR,X1,X2,...)
*
* W.JUENGST    (1/86)
*
        AORG >FD54
REG     DATA 0,0,0
BUF     DATA 0,0,0,0,0,0,0,0,0,0,0,0,0
SCTAD   DATA >3CEF
FLAG    DATA 0
XWSECT  LWPI REG
        CLR  @FLAG
        JMP  GOON
XRSECT  LWPI REG
        SETO @FLAG
GOON    CLR  0
        LI   1,1
        BLWP @>200C
        BLWP @>2018
        DATA >12B8
        MOV  @>834A,3
        LI   1,2
        BLWP @>200C
        BLWP @>2018
        DATA >12B8
        MOV  @>834A,@>8350
        MOV  @FLAG,@FLAG
        JEQ  WRT
        AI   3,>100
WRT     SWPB 3
        MOV  3,@>834C
        MOV  @SCTAD,@>834E
        LWPI >83E0
        LI   12,>1100
        SBO  0
        MOV  @>400A,1
        MOV  @2(1),1
        BL   *1
        SBZ  0
        SBZ  0
RET     SB   @>837C,@>837C
        B    @>70
PEEKV   LWPI REG
        BL   @GETVAL
        MOV  @>834A,0
        LI   1,BUF
        MOVB @>8312,2
        SRL  2,8
        DEC  2
        BLWP @>202C
        CLR  0
NA      CLR  @>834A
        MOVB @BUF-1(2),@>824B
        BLWP @>2018
        DATA >20
        MOV  2,1
        INC  1
        BLWP @>2008
        DEC  2
        JNE  NA
R       LWPI >83E0
        JMP  RET
POKEV   LWPI REG
        BL   @GETVAL
        MOV  @>834A,12
        CLR  0
        MOVB @>8312,13
        SRL  13,8
        MOV  13,1
        DEC  13
PI      BL   @GETVA
        MOVB @>834B,@BUF-2(1)
        DEC  1
        CI   1,1
        JNE  P1
        LI   1,BUF
        MOV  12,0
        MOV  13,2
        BLWP @>2024
        CLR  0
        JMP  R
GETVAL  LI   1,1
GETVA   CLR  0
        BLWP @>200C
        BLWP @>2018
        DATA >12B8
        RT
        END
```

DAS GEHEIMNIS VON CHICHEN ITZA

Ihre Aufgabe in diesem Action-Grafik-Adventure ist es, die geheime Grabkammer des Gottes "Chaca" und dessen Schatz zu finden. Ein bißchen Glück, aber vor allem Geschicklichkeit, sind für die Bewältigung des Adventures nötig.
Das Programm ist in drei Teile aufgeteilt, so daß nur ein Recorder und das Ext. Basic-Modul erforderlich sind.
Teil 1 enthält neben dem Titelbild und Char-Definitionen für Teil 2 die Hintergrundgeschichte des Adventures und eine Bedienungsanleitung.
Teil 2 und 3 sind dann das eigentliche Spiel.
Wenn vom Menü des Teils 1 aus durch Drücken der "3" das Spiel begonnen werden soll (Laden von Teil 2), so sehen Sie einen Sprite auf dem Bildschirm nach oben laufen. Drücken Sie nun solange die ENTER-Taste, bis dieser stillsteht. Stellen Sie nun Ihren Recorder auf "Play". Ist Teil 2 geladen, so bewegt sich der Sprite wieder. Drücken Sie jetzt nochmals die ENTER-Taste, so beginnt das Spiel.
Ebenso ist beim Laden des Teils 3 von Teil 2 aus (falls dieser erfolgreich geschafft wurde!) zu verfahren. Falls Sie in Bild 1 oder 2 einen Fehler machen, beginnt alles von



Gute Grafik und netter Sound zeichnen dieses Adventure aus

vorn. Erreichen Sie jedoch Bild 3, so fangen Sie bei einem Fehler immer wieder in der untersten Ebene dieses Bildes an, es sei denn, der Fehler erfolgt in Ebene 1.
Steuerung: Die Steuerung erfolgt mit Joystick 1 in üblicher Weise. Der Feuerknopf wird erst ab Bild 3, Ebene 2 benötigt, um nach links, bzw. rechts zu springen (statt zu gehen).
Anmerkung des Autors: „Ich übernehme keine Haftung für verbogene und an die Wand geschmissene Joysticks (Ende des Zitats)"!

INTEGRAL-RECHNUNG

Nach dem Eingeben und Starten des Programms in den TI mit Extended Basic, meldet sich der Computer mit dem Menü. Unter Punkt 1 kann man Kurzinformationen erhalten, die eigentlich die Benutzung des Programms gestatten. Im Normalfall wird der User die Option STETIG verwenden, da in den meisten Fällen die zu integrierende Funktion bekannt ist. Bei dieser Option muß man den Rechner stoppen und die Funktion in Zeile 260 definieren. Nach dem Neustart und der Programmwahl 2 fragt der Rechner zuerst die Intervallgrenzen ab, zwischen denen integriert werden soll. Nun muß noch die Zahl der Subintegrale eingegeben werden (nur gerade Anzahl möglich). Von dieser Zahl hängt die Genauigkeit und Schnelligkeit der Berechnung ab (je größer, desto genauer, aber auch um so langsamer). Nach dieser Eingabe berechnet der Computer das Integral und zeigt das Ergebnis an. Sollte man eine Wiederholung wünschen, so gebe man FCTN REDO ein, zum Menü zurück kommt man mit FCNT BACK.
Beim Programmteil UNSTETIG ist die Funktion nicht in der Form F(x)=... bekannt. Man kennt nur einige Punkte, durch welche die Funktion geht. Diese Punkte müssen den gleichen Abstand zwischeneinander besitzen. Dieser Abstand wird zuerst vom Computer erfragt. Anschließend erkundigt er sich wieder nach der Anzahl der Subintegrale (wieder nur gerade Anzahl möglich). Dabei ist zu beachten, daß N+1 Punkte der Funktion bekannt sein müssen (F(0), F(1), F(2),... F(N)). Nun folgt die Eingabe der Punkte. Sollten einer oder mehrere falsch eingegeben worden sein, so besteht noch die Möglichkeit der Korrektur.
Ist der Eingabeteil abgeschlossen, berechnet der Computer das Integral.
Auch bei dieser Option ist die Möglichkeit der Wiederholung gegeben (wie oben).

LISTINGS

```
10 ! ***********************
11 ! * DAS GEHEIMNIS  VON *
12 ! *    CHICHEN ITZA    *
13 ! *      (Teil 1)      *
14 ! *                    *
15 ! *    Copyright by    *
16 ! *                    *
17 ! *  Manfred Lipowski  *
18 ! *                    *
19 ! * Benoetigte Geraete *
20 ! *  TI99/4A  Konsole  *
21 ! *     Ext. Basic     *
22 ! *     Joystick 1     *
23 ! *    Cassettenrec.   *
24 ! *(oder DISK+32K-Erw.)*
25 ! *                    *
26 ! *  Speicherbelegung  *
27 ! *    10910 Bytes     *
28 ! *                    *
29 ! ***********************
100 CALL CLEAR :: CALL SCREE
N(2):: CALL MAGNIFY(3):: CAL
L COLOR(2,16,1,14,16,1,10,5,
1,9,14,1,12,11,1,1,4,1,3,7,1
,11,4,1)
110 DATA 40,F04848494A4A4AF1
00000086484442AC000000444529
2910000000CA2D2929C9,44,7088
88A1BAAB8A711010109C52D21292
0004006495F5856500000 0A55654
5454
120 DATA 137,00200026A8A4A2A
C,96,3C6642404042663CA5E7427
E4242E7A5181800181818 3C663C6
640707840663C
130 DATA 100,A5E7725A4E46E7A
5FF7E181818183C66FF7E060C183
07EFF3C66427E6642E7A5,104,65
555562525262002 2362A2A2222A2
0022202222222 2BA00C4AAAACA8A
8A84
140 DATA 108,888989A8A8AA510
0C92A0C8A494989004102425942 4
241009E50508C42528C,120,7FC0
808F989090919190909 88F80C07F
FE0301F1190909898909 0919F101
03FE
150 DATA 124,91998999B1A1B19
1FF0070BB0E0000FF,33,FFFFFFF
FFFFFFFFF,112,0000000030F3F
FF030F3FFFFFFFFFFFC0F0FCFFFF
FFFFFF00000000C0F0FCFF
160 DATA 116,FFFFFFFFFFFCF0C
0FFFCF0C,118,FF3F0F03,119,FF
FFFFFFFF3F0F03,36,FFFFFFFFFF
FCF0C,37,,38,FFFCF0C,39,,48,
FF3F0F03
170 DATA 49,,50,FFFFFFFFFF3F
0F03,51,,52,000070FCFEFF7F3F
00000E3F7FFFFEFC,56,03040A08
05344A4B4A281E03,58,C0205010
A02C52D2521478C
180 DATA 60,03070F0F07377F7F
7F3F1F03,62,C0E0F0F0E0ECFEFE
FEFCF8C,136,FFFF
190 DATA 128,071820211E0C182
06649C1827C55AA7CE0180C02424
20202845820808,132,071F3F3F1
F0F1F3F7F7FFFFF7F7FFE7CE0F8F
CFEFEFEFEFEFC7820808
200 RESTORE 110 :: FOR A=1 T
O 30 :: READ B,B$ :: CALL CH
AR(B,B$):: NEXT A
210 CALL HCHAR(1,1,125,32)::
 CALL VCHAR(1,1,124,24):: CA
LL HCHAR(24,1,125,32):: CALL
 VCHAR(1,32,124,24)
220 CALL HCHAR(1,1,120):: CA
LL HCHAR(1,32,122):: CALL HC
HAR(24,1,121):: CALL HCHAR(2
4,32,123)
230 DISPLAY AT(3,9):"()  ,-./
  *+" :: DISPLAY AT(5,4):"`
a b ` a c d  b e f g" :: CAL
L HCHAR(3,18,137)
240 FOR A=104 TO 111 :: CALL
 HCHAR(23,12+A-103,A):: NEXT
 A
250 CALL VCHAR(13,16,33,6)::
 CALL VCHAR(13,17,33,6):: CA
LL HCHAR(11,14,33,6):: DISPL
AY AT(12,12):"vw!!tu" :: DIS
PLAY AT(19,12):"pq!!rs"
260 CALL HCHAR(20,9,136,16):
: DISPLAY AT(10,14):"45" ::
CALL SPRITE(#1,128,2,137,121
,#2,132,16,137,121)
270 CALL SPRITE(#3,56,11,66,
121,#4,60,5,66,121,#5,48,13,
88,113,#6,36,13,88,129)
280 DISPLAY AT(21,4):"PRESS
ANY KEY TO BEGIN" :: CALL CO
LOR(5,8,1,6,8,1,7,8,1,8,8,1)
290 CALL COLOR(#4,7,#3,5,#4,
9,#3,6,#4,10,#3,8):: CALL KE
Y(0,K,S):: IF S=0 THEN CALL
COLOR(#4,9,#3,2,#4,7):: GOTO
 290
300 CALL DELSPRITE(ALL)
310 CALL CLEAR :: CALL CHARS
ET :: RANDOMIZE :: FOR A=65
TO 90 :: CALL CHARPAT(A,B$):
: CALL CHAR(A+32,B$):: NEXT
A
320 CALL CHAR(64,"3C4299A1A1
99423C",128,"00014241E242616
06354595241404040E0F0B0F0604
CBA64ACF24AF1A2C28C88")
330 CALL CHAR(132,"070F0D0F0
6325D26354F528F4543311100804
28247428606C62A9A4A82020202"
,136,"4245444344444848505048
4844464C58A0D09060101020204 0
40202010183060")
340 CALL CHAR(40,"000000FF",
140,"050B0906080804040202040
```

```
408180C0642A222C222221212 0A0
A12122262321A")
350 CALL COLOR(2,11,1,3,5,1,
4,5,1,5,3,1,6,3,1,7,3,1,8,3,
1,9,8,1,10,8,1,11,8,1,12,8,1
)
360 CALL COLOR(0,7,1,1,16,1)
:: DISPLAY AT(1,3):"MANFRED
LIPOWSKI PRESENT": : :".....
das geheimnis von": :"......
.chichen itza !"
370 DISPLAY AT(8,3):"@ 1985
OKTOBER M.LIPOWSKI" :: CALL
SPRITE(#1,128,15,24,36,#2,13
2,15,24,200,#3,136,15,40,36,
#4,140,15,40,200)
380 CALL HCHAR(10,1,40,32)::
 DISPLAY AT(12,3):"1 = EINFU
EHRUNG": :"..2 = SPIELERKLAE
RUNG": :"..3 = SPIELANFANG"
390 DISPLAY AT(18,1):: DISPL
AY AT(20,1):: DISPLAY AT(22,
1):: DISPLAY AT(24,1)BEEP
400 CALL Z(1,K):: IF K<49 OR
 K>51 THEN 400 ELSE ON K-48
GOTO 410,570,780
410 DISPLAY AT(12,3):"die st
adt chichen itza": :"..,ceno
te von den indios": :"..und
dzonot von den mayas": :"..g
enannt,liegt in einem"
420 DISPLAY AT(20,3):"karsti
gen gebiet oestlich": :"..vo
n mexico,auf der halb-": :".
.insel yutacan." :: CALL Z(2
,K)
430 DISPLAY AT(12,3):"die la
ndschaft dort ist": :"..meis
t steinig und nur": :"..spae
rlich bewachsen.": :"..wasse
r loest das kalkge-"
440 DISPLAY AT(20,3):"stein
und verursacht rit-": :"..ze
n,durch die das wasser": :".
.in hoehlen sickert." :: CAL
L Z(2,K)
450 DISPLAY AT(12,3):"durch
korrosion werden die": :"..h
oehlen,auch dolinen ge-": :"
..nannt,immer groesser.die":
 :"..geologen sind so gut wi
e"
460 DISPLAY AT(20,3):"sicher
,dass der heilige": :"..brun
nen von chichen itza": :"..s
o entstanden ist." :: CALL Z
(2,K)
470 DISPLAY AT(12,3):"die ma
yas hatten einen": :"..regen
gott namens chaca.um": :"..s
ich seine gunst zu er-": :".
.halten erklaerten sie die"
480 DISPLAY AT(20,3):"brunne
n zu heiligen staet-": :"..t
en und opferten dort was": :
"..ihnen lieb und teuer war.
" :: CALL Z(2,K)
490 DISPLAY AT(12,3):"gold,j
uwelen und auch men-": :"..s
chen wurden in chichen": :".
.itza geopfert.gold und": :"
..edelsteine lockten seit"
500 DISPLAY AT(20,3):"jahrhu
nderten alle moeg-": :"..lic
hen abenteurer an. die": :".
.ersten waren spanier." :: C
ALL Z(2,K)
510 DISPLAY AT(12,3):"nach d
er unterwerfung des": :"..az
tekenreiches versuchten": :"
..sie mit primitiven mitteln
": :"..das gold zu bergen."
520 DISPLAY AT(20,3):"als di
e suche zu schwierig": :"..w
urde,geriet der heilige": :"
..brunnen in vergessenheit."
 :: CALL Z(2,K)
530 DISPLAY AT(12,3):"zwisch
en 1904 und 1907 ge-": :"..l
ang es einer amerikani-": :"
..schen expedition,gold-": :
"..schmuck und juwelen zu"
540 DISPLAY AT(20,3):"bergen
.zwischen 1954 und": :"..197
2 barg eine mexikanisch": :"
..-amerikanische expedition"
 :: CALL Z(2,K)
550 DISPLAY AT(12,3):"mehr a
ls 4000 stuecke.bei": :"..di
eser expedition ereigne-": :
"..ten sich mysterioese vor-
": :"..faelle:leiter,cheftau
cher"
560 DISPLAY AT(20,3):"und di
rektor des instituts": :"..s
tarben eines unnatuerli-": :
"..chen todes..." :: CALL Z(
2,K):: GOTO 380
570 DISPLAY AT(12,3):"ihren
namen bitte": :"..herr arche
ologe.": :"..DR.:": :"": :"
": :""
580 ACCEPT AT(16,7)BEEP VALI
DATE(UALPHA,". -")SIZE(14):N
AM$ :: CALL FGP(NAM$)
590 FOR A=1 TO 450 :: NEXT A
600 DISPLAY AT(12,3):"SIE SI
ND archeologe UND": :"..WOLL
EN DIE GEHEIME grab-": :"..k
ammer DES ALTEN maya": :"..g
ottes chaca IN chichen"
610 DISPLAY AT(20,3):"itza F
INDEN." :: CALL Z(2,K)
620 DISPLAY AT(12,3):"DAZU B
ENOETIGEN SIE ERST-": :"..MA
L DIE sonnenscheibe DIE": :"
```



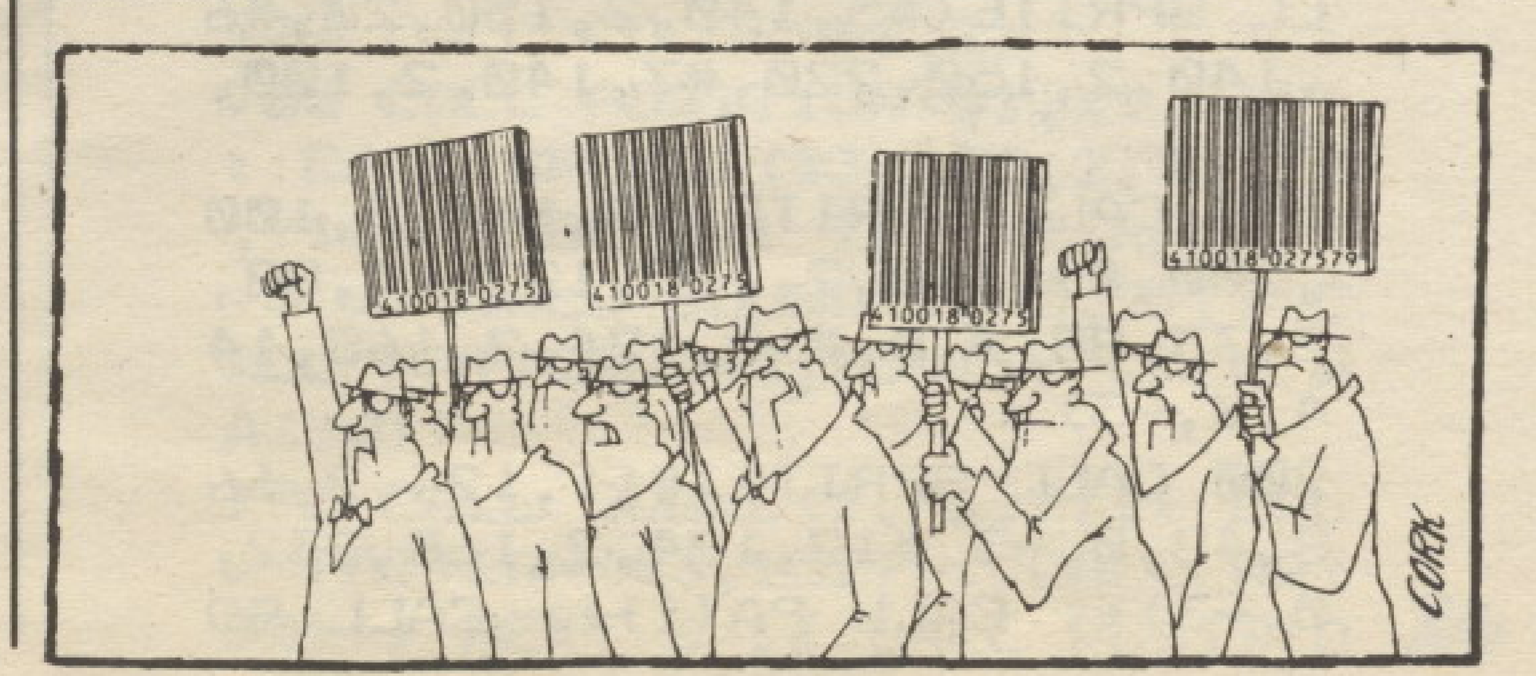
```
..DIE mayas VOR LANGER ZEIT"
: :"..IM brunnen VERSENKTEN.
"
630 DISPLAY AT(20,3):"ABER V
ORSICHT BEIM tauchen": :"..A
UF DEM GRUND DES BRUNNENS":
:"..SIND EINIGE muraenen." :
: CALL Z(2,K)
640 DISPLAY AT(12,3):"EIN ei
nziger biss IST": :"..toedli
ch.WERDEN SIE BEIM": :"..TAU
CHEN gelb SOLLTEN SIE": :"..
SO SCHNELL WIE MOEGLICH"
650 DISPLAY AT(20,3):"NACH o
ben SCHWIMMEN UM": :"..luft
ZU SCHNAPPEN.VERSAEU-": :"..
MEN SIE DIES sterben SIE !"
:: CALL Z(2,K)
660 DISPLAY AT(12,3):"DIE so
nnenscheibe LIEGT": :"..IRGE
NDWO AUF DEM GRUND DES": :".
.BRUNNENS-schwarz-.": :"..HA
BEN SIE SIE ALSO GEFUN-"
670 DISPLAY AT(20,3):"DEN ER
TOENT EIN akusti-": :"..sche
s signal.BEGEBEN SIE": :"..S
ICH IN RICHTUNG leiter !" ::
 CALL Z(2,K)
680 DISPLAY AT(12,3):"JETZT
MUESSEN SIE DEN": :"..RICHTI
GEN tempel FINDEN.": :"..SIE
 HABEN nur 2 chanchen": :"..
IHN ZU FINDEN.VERSAGEN"
690 DISPLAY AT(20,3):"SIE HO
LT SIE DER ALTE": :"..mayago
tt chaca ! ! !" :: DISPLAY A
T(24,3):: CALL Z(2,K)
700 DISPLAY AT(12,3):"HABEN
SIE DEN RICHTIGEN": :"..temp
el GEFUNDEN WIRD IHNEN": :".
.DER GOTT EINEN ZUGANG ZUR":
 :"..grabkammer ZEIGEN."
710 DISPLAY AT(20,3):"EINE D
ER unterwasserhoeh-": :"..le
n WIRD DURCH DEN LICHT-": :"
..EINFALL AUF DIE sonnen-" :
: CALL Z(2,K)
720 DISPLAY AT(12,3):"scheib
e ERLEUCHTET.DORT-": :"..HIN
 MUESSEN SIE tauchen.": :"..
vorsicht VOR DEN ANDEREN": :
"..HOEHLEN.ETWAS grauenvolle
s"
730 DISPLAY AT(20,3):"ERWART
ET SIE DARIN ! WENN": :"..SI
E DURCH DIE HOEHLE TAU-": :"
..CHEN PASSEN SIE AUF." :: C
ALL Z(2,K)
740 DISPLAY AT(12,3):"VERLET
ZEN SIE SICH NICHT": :"..AN
DEN felskannten SONST": :"..
hauchen SIE IHR leben AUS.":
 :"..VERSUCHEN SIE DIE sonne
n"
750 DISPLAY AT(20,3):"scheib
e AUF DER LINKEN": :"..SEITE
 ZU ERREICHEN." :: DISPLAY A
T(24,3):: CALL Z(2,K)
760 DISPLAY AT(12,3):"DANACH
 MUESSEN SIE NOCH": :"..EINI
GE gefahren AUF SICH": :"..N
EHMEN UM DEN schatz ZU": :".
.ERREICHEN.VIEL glueck"
770 DISPLAY AT(20,3):"DR.";N
AM$;" !" :: DISPLAY AT(22,3)
:: DISPLAY AT(24,3):: CALL Z
(2,K):: GOTO 380
780 DISPLAY AT(12,3):"erster
 piepton: cassette": :"..bis
 listing 2 vorspulen.": :"..
zweiter ton:play druecken.":
 :"..dann enter druecken"
790 DISPLAY AT(20,3):"3,4ter
 ton:enter druecken": :"..al
les klar ?": :"..bitte taste
 druecken." :: CALL Z(2,K)
800 DISPLAY AT(24,3):"BITTE
WARTEN.LESE DATEN." :: FOR B
=1 TO 400 :: NEXT B :: CALL
CLEAR
810 CALL MOTION(#3,-5,0,#1,-
5,0,#2,5,0,#4,5,0)
820 DATA 32,,33,080808,34,00
0000000808080808080808080808
08,36,0808101028240808000000
080808000000808040408100808 08
21442412520208
830 DATA 40,FF8090A890878485
FF00251B00FF00FFFF01091509E1
21A1FF00A4D800FF00FF,44,8585
858585858585A1A1A1A1A1A1A1A1
804020100F0F0F0F0F0F0F0F0F0F
0F0F
840 DATA 48,01020408F0F0F0F0
F0F0F0F0F0F0F0F0FD050B0A0B14
1714282F28505F50A0FF,52,BFA0
D050D028E82814F4140AFA0A05FF
FF00FF00FF00FF0000FF0000FF00
00FF
850 DATA 56,FFFFFFFFFFFFFFFF
0101030307070F0F1F1F3F3F7F7F
FFFF8080C0C0E0E0F0F,60,F8F8F
CFCFEFEFFFF01030102030F3CFB8
0C08040C0F03CDF
860 DATA 64,427E427E427E427E
,65,,66,10101010080808080808
08081010101,68,1010101,69,08
08080000000008,70,0808101020
20100808040404040081008
870 DATA 72,,73,AA55AA55AA55
AA55,74,FFFFFFFFFFFFFFFF,80,
AA55AA55AA55AA55FFFFFFFFFFFF
FFFF
880 DATA 88,181818247EFFBD99
BDBD242424242442,90,,91,,116
,000000000000107030102,118,00
```

L I S T I N G S

```
000000800080E0C08
890 DATA 120,387CFEFE7C38101
01010101010,122,,123,,124,00
01076EF83,125,,126,00C0F03C0
E03,127,
900 DATA 128,0000001FA14F10A
041,130,00E0130BBDFEF810E,13
2,0307050F0F0707070A152A2924
130A05C0E0A0F0F0E0E0E0D028B4
54A448502
910 DATA 136,0307070D0F07060
5070A354992954A24C0E0E0B0F0E
060A0E050AC92492952A4,140,04
0E3F7FFFFFFF7F7F3F3F3F7F7F3E
1C38FCFEFFFEFCFCFCFEFFFFFEFC
FCF06
920 DATA 76,000000407070F1F
0F070F0501010000000040E0E0F0
F8F8F0E0E0C04,84,00082D7F7F3
F1F3F7F7F3F1F0F1F1C0000D8FCF
EFCF8FCFEFEFEFCFCF8381
930 RESTORE 820 :: FOR A=1 T
O 39 :: READ B,B$ :: IF B=12
8 THEN CALL DELSPRITE(ALL)
940 CALL CHAR(B,B$):: NEXT A
 :: CALL MAGNIFY(4):: CALL S
PRITE(#1,88,15,190,122,-4,0)
950 FOR A=0 TO 14 :: CALL CO
LOR(A,1,1):: NEXT A
960 RUN "CS1"
970 SUB Z(X,K):: ON X GOTO 1
000,1040
980 DATA 784,784,740,784,880
,784,740,659,784,784,659,784
990 DATA 196,247,262,311,294
,262,247,208,196
1000 RESTORE 980 :: FOR B=1
TO 11 :: READ C
1010 CALL SOUND(-999,C,30,C,
30,C,30,-4,B)
1020 NEXT B :: CALL KEY(0,K,
S):: IF S<>0 THEN SUBEXIT EL
SE 1000
1030 CALL PATTERN(#1,132,#3,
140,#2,128,#4,136)
1040 RESTORE 990 :: FOR A=1
TO 9 :: READ B :: CALL SOUND
(-999,B,21,B+2,22,B*2,30,-4,
A*2):: NEXT A :: CALL PATTER
N(#1,128,#3,136,#2,132,#4,14
0)
1050 CALL KEY(0,K,S):: IF S<
>0 THEN SUBEXIT
1060 RESTORE 990 :: FOR A=1
TO 9 :: READ B :: CALL SOUND
(-999,B,30,B,30,B*2,30,-4,A*
2):: NEXT A
1070 CALL KEY(0,K,S):: IF S<
>0 THEN SUBEXIT ELSE 1030
1080 SUBEND
1090 SUB FGP(NAM$):: IF NAM$
="JONES" OR NAM$="INDIANA JO
NES" OR NAM$="INDI" OR NAM$=
"INDIANA" THEN 1110 ELSE 110
0
1100 DISPLAY AT(18,3):"DANKE
 DR.";NAM$;" !" :: CALL SOUN
D(99,196,0):: CALL SOUND(99,
262,0):: SUBEXIT
1110 DATA 3,330,3,349,5,392,
8,523,3,294,3,330,6,349
1120 DATA 3,392,3,440,4,494,
8,698,3,440,3,494,4,523,6,39
2
1130 DISPLAY AT(18,3):"DANKE
 DR.";NAM$;" !"
1140 RESTORE 1110 :: FOR A=1
 TO 15 :: READ B,C :: B=B*99
 :: CALL SOUND(B,C,0,C/2,0,C
+2,1):: NEXT A
1150 SUBEND
```

LISTINGKORREKTUR FÜR TI-REVUE 7/86

Sorry, bei der Bearbeitung der Listings für die letzte Ausgabe waren unsere Programmierer anscheinend schon in Urlaubsstimmung. Daß sich dadurch auch fehlerhafte Listings einschleichen konnten, blieb natürlich nicht aus.
Hier die Korrektur:
Im Programm „Kassenbuch" ändert bitte den Sprungbefehl in Zeile 950 von GOTO 32767 in GOTO 1000.
Der nächste Lapsus ist leider nicht mit einem Satz aus der Welt geschafft.
Im Programm „Trapper" hat sich ein schwerwiegender Denkfehler eingeschlichen, der den ganzen Ablauf des Spieles verfälscht.
Liefert der Trapper zum ersten Mal seine Felle im Depot ab, ist die Welt noch vollkommen in Ordnung. Sobald er jedoch wieder ein gutes Geschäft gemacht hat und die nächsten Felle lagern will, wird die erste Variable überschrieben und der Depotinhalt ist weg. Dem Trapper verbleiben damit nur noch die Felle aus der letzten Lieferung und die nichteingelagerten Felle.
Um den Trapper vor derartigen unkalkulierbaren Verlusten zu bewahren, müssen die Variablen für die Einlagerung und die Lagerhaltung getrennt werden. Der Vorgang bleibt für alle vier Fellarten der gleiche und sei hier kurz für die Lagerung der Nerzfelle beschrieben:
Die numerischen Variablen "NFAB" in Zeile 4440–4470 ändern in "NFXY".
Jetzt wird eine zusätzliche Zeile zwischen 4470 und 4480 eingefügt, in der die abgelegten Felle zum Lagerinhalt addiert werden, also "4476 NFAB = NFAB+NFXY"
Damit gehen dem Trapper schon einmal keine Nerzfelle mehr verloren. Sinngemäß muß dieser Vorgang mit den Variablen für die Luchs-, Fuchs- und Hasenfelle wiederholt werden und das Programm läuft korrekt.



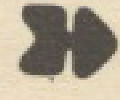
LISTINGS

```
10 ! **********************
11 ! * DAS GEHEIMNIS  VON *
12 ! *    CHICHEN ITZA    *
13 ! *      (Teil 2)      *
14 ! *                    *
15 ! *    Copyright by    *
16 ! *                    *
17 ! *  Manfred Lipowski  *
18 ! *                    *
19 ! * Benoetigte Geraete *
20 ! *  TI99/4A  Konsole  *
21 ! *     Ext. Basic     *
22 ! *     Joystick 1     *
23 ! *    Cassettenrec.   *
24 ! *(oder DISK+32K-Erw.)*
25 ! *                    *
26 ! *  Speicherbelegung  *
27 ! *    12590 Bytes     *
28 ! *                    *
29 ! **********************
100 CALL CLEAR :: RANDOMIZE
:: CALL MAGNIFY(3):: CALL DE
LSPRITE(ALL)
110 DATA "!#()+*$&","!´,.0-!
#","$#,/1-%´","9826648;",":8
37758<"
120 FOR A=1 TO 5 :: FOR B=1
TO 32 :: CALL HCHAR(A,B,INT(
RND*8)+32):: NEXT B :: NEXT
A
130 RESTORE 110 :: FOR A=1 T
O 5 :: READ A$ :: DISPLAY AT
(A+1,0)SIZE(8):A$ :: DISPLAY
 AT(A+1,11)SIZE(8):A$ :: DIS
PLAY AT(A+1,21)SIZE(8):A$
140 NEXT A :: FOR A=7 TO 10
:: FOR B=1 TO 32 :: CALL HCH
AR(A,B,INT(RND*7)+65):: NEXT
 B :: NEXT A
150 CALL VCHAR(7,11,64,4)::
CALL VCHAR(7,22,64,4):: CALL
 HCHAR(11,1,72,96):: CALL HC
HAR(14,1,73,96):: CALL HCHAR
(17,1,74,96)
160 CALL DELSPRITE(#1,#3)::
CALL HCHAR(1,6,37,2):: CALL
HCHAR(1,16,34,2):: CALL HCHA
R(1,26,33,2)
170 CALL COLOR(1,8,13,2,5,15
,3,5,15,4,15,13,5,7,11,6,5,6
,7,2,5):: CALL SCREEN(13)
180 CALL HCHAR(20,1,80,96)::
 CALL HCHAR(23,1,81,64):: CA
LL SPRITE(#5,140,2,150,20,#6
,140,2,150,220,#7,140,2,100,
40)
190 CALL SPRITE(#8,140,2,100
,200,#9,140,2,125,122,#1,88,
2,33,88,0,0,#10,124,2,168,14
4,0,-3)
200 CALL SPRITE(#11,124,2,16
8,48,0,-3,#12,124,2,168,232,
0,-3):: CALL PA(1):: CALL SO
(2)
210 SON,TE1,TOL=0 :: LOC=28
220 SO1=INT(RND*210)+20 :: T
EM=INT(RND*3)+1
230 CALL PATTERN(#1,88)
240 CALL JOYST(1,X,Y):: CALL
 POSITION(#1,X1,Y1)
250 IF X=4 THEN 260 ELSE IF
X=-4 THEN 270 ELSE IF Y=4 TH
EN 280 ELSE IF Y=-4 THEN 290
 ELSE 230
260 Y1=Y1+8 :: PAT=100 :: TO
N=-5 :: GOTO 300
270 Y1=Y1-8 :: PAT=92 :: TON
=-6 :: GOTO 300
280 X1=X1-8 :: PAT=108 :: TO
N=-7 :: GOTO 300
290 X1=X1+8 :: PAT=108 :: TO
N=-7
300 IF Y1<=8 THEN Y1=248 ELS
E IF Y1>=248 THEN Y1=8
310 CALL PATTERN(#1,PAT):: C
ALL SOUND(-5,TON,5):: CALL G
CHAR(X1/8+2,Y1/8+1,W):: IF W
<>32 THEN 320 ELSE 480
320 IF W>=33 AND W<=39 THEN
CALL SOUND(300,-1,0):: GOTO
230
330 IF W>=64 AND W<=71 THEN
GOTO 490
340 IF W=54 OR W=55 THEN 480
 ELSE IF W=47 OR W=49 THEN 3
60
350 IF W>=40 AND W<=60 AND Y
=4 THEN CALL SOUND(300,-2,0)
:: GOTO 230 ELSE 480
360 CALL LOCATE(#1,X1,Y1)::
IF SON=0 THEN 370 ELSE 430
370 CALL PATTERN(#1,88):: CA
LL SOUND(2,-6,0):: CALL WA(I
NT(RND*999)+333):: CALL COLO
R(#1,1):: B=INT(RND*5)+5 ::
CALL POSITION(#B,X2,Y2):: CA
LL LOCATE(#1,X2,Y2)
380 CALL WA(INT(RND*1500)+30
0)
390 CALL DELSPRITE(#3):: CAL
L PATTERN(#1,128):: CALL COL
OR(#1,2):: CALL MOTION(#1,-3
,INT(RND*4)-2):: CALL SOUND(
1,-5,0)
400 CALL POSITION(#1,X1,Y1):
: IF X1<=80 THEN 410 ELSE 40
0
410 CALL POSITION(#1,X1,Y1):
: CALL LOCATE(#1,80,Y1)
420 CALL MOTION(#1,0,INT(RND
*4)-2):: :: CALL SO(1):: CAL
L WA(1250):: GOTO 160
430 TE1=TE1+1 :: IF TE1=3 TH
EN 370
440 CALL POSITION(#1,X1,Y1):
: ON TEM GOTO 450,460,470
```

```
450 IF Y1<64 THEN A=6 :: GOT
O 800 ELSE 230
460 IF Y1>96 AND Y1<152 THEN
 A=16 :: GOTO 800 ELSE 230
470 IF Y1>160 THEN A=26 :: G
OTO 800 ELSE 230
480 CALL LOCATE(#1,X1,Y1)::
CALL PATTERN(#1,PAT+4):: CAL
L SOUND(-5,TON,3):: GOTO 240
490 CALL PATTERN(#1,88):: CA
LL PA(2):: FOR A=1 TO INT(RN
D*5)+3 :: CALL PATTERN(#1,88
):: CALL SOUND(1,-5,9):: CAL
L WA(50):: CALL PATTERN(#1,1
08)
500 CALL SOUND(1,-6,6):: CAL
L WA(50):: NEXT A :: CALL MO
TION(#1,-4,0):: CALL WA(125)
:: CALL MOTION(#1,5,0):: CAL
L PATTERN(#1,112)
510 CALL POSITION(#1,X1,Y1):
: IF X1>=64 THEN 520 ELSE 51
0
520 FOR A=0 TO 30 STEP 2 ::
CALL SOUND(-99,-5,A):: NEXT
A :: CALL MOTION(#1,0,0):: C
ALL PATTERN(#1,100):: PAT=10
0 :: LUF=40
530 LUF=LUF-1 :: CALL PATTER
N(#1,PAT):: CALL COINC(ALL,Z
):: IF Z THEN 700 ELSE CALL
MU
540 IF SON=1 THEN 550 ELSE C
ALL COINC(#1,179,S01,8,I)::
IF I THEN CALL MOTION(#1,0,0
):: SON=1 :: CALL SOUND(99,1
96,0):: CALL SOUND(99,262,0)
550 IF LUF=18 THEN CALL COLO
R(#1,11):: CALL SOUND(-300,-
1,0)
560 CALL JOYST(1,X,Y):: IF X
=-4 THEN PAT=92 ELSE IF X=4
THEN PAT=100 ELSE PAT=PAT
570 CALL PATTERN(#1,PAT+4)::
 CALL MOTION(#1,-Y*1,X*2)::
CALL COINC(ALL,Z):: IF Z=-1
THEN 700
580 IF LUF<=0 THEN CALL SOUN
D(-300,-4,0):: GOTO 390
590 CALL POSITION(#1,X1,Y1):
: IF X1<=78 THEN 600 ELSE 61
0
600 CALL MOTION(#1,0,0):: CA
LL LOCATE(#1,81,Y1):: LUF=40
 :: CALL COLOR(#1,2):: CALL
SOUND(-9,-5,4):: CALL SOUND(
-9,-5,9):: GOTO 640
610 IF X1>186 THEN 620 ELSE
530
620 CALL MOTION(#1,-11,0)::
FOR A=0 TO 30 STEP 5 :: CALL
 SOUND(-99,-7,A):: NEXT A ::
 GOTO 530
```

LISTINGS

```
630 GOTO 530
640 CALL POSITION(#1,X1,Y1):
: CALL MOTION(#1,0,0):: CALL
 LOCATE(#1,80,Y1):: IF Y1>=7
8 AND Y1<=92 THEN A=80 :: GO
TO 660 ELSE IF Y1>=166 AND Y
1<=176 THEN A=168 :: GOTO 66
0
650 FOR A=10 TO 50 STEP 10 :
: CALL SOUND(-99,880-A*2,4,4
40-A*2,5):: NEXT A :: GOTO 5
30
660 CALL SOUND(-9,-6,9):: CA
LL PATTERN(#1,108):: CALL SO
UND(-9,-7,11):: CALL LOCATE(
#1,80,A):: CALL PA(1):: CALL
 SOUND(-9,-6,14)
670 FOR A=1 TO 6 :: CALL POS
ITION(#1,X1,Y1):: CALL SOUND
(1,-5,A*3):: CALL PATTERN(#1
,108):: CALL WA(30):: CALL L
OCATE(#1,X1-8,Y1):: CALL PAT
TERN(#1,112)
680 CALL SOUND(1,-7,A*3):: C
ALL WA(30):: NEXT A :: CALL
POSITION(#1,X1,Y1):: CALL LO
CATE(#1,X1+1,Y1):: GOTO 230
690 GOTO 530
700 CALL MOTION(#1,0,0):: CA
LL SOUND(-35,-7,9)
710 CALL COINC(#1,#10,10,C):
: IF C THEN 760 ELSE CALL CO
INC(#1,#11,10,C):: IF C THEN
 760 ELSE CALL COINC(#1,#12,
10,C):: IF C THEN 760
720 CALL COINC(#1,#LOC,15,N)
:: IF N THEN 870
730 CALL COINC(#1,#5,20,C)::
 CALL COINC(#1,#6,20,C1):: C
ALL COINC(#1,#7,20,C2):: CAL
L COINC(#1,#8,20,C3):: CALL
COINC(#1,#9,20,C4)
740 IF C THEN A=5 :: GOTO 77
0 ELSE IF C1 THEN A=6 :: GOT
O 770 ELSE IF C2 THEN A=7 ::
 GOTO 770 ELSE IF C3 THEN A=
8 :: GOTO 770 ELSE IF C4 THE
N A=9 :: GOTO 770
750 GOTO 530
760 CALL COLOR(#1,2):: CALL
SOUND(5,-5,3):: CALL SOUND(5
,-6,6):: CALL SOUND(5,-7,9):
: GOTO 390
770 CALL COLOR(#1,2)
780 CALL POSITION(#1,X1,Y1):
: CALL SPRITE(#3,132,2,X1-10
,Y1):: FOR B=1 TO INT(RND*9)
+3 :: CALL SOUND(1,-5,2):: C
ALL PATTERN(#3,136):: CALL W
A(30)
790 CALL SOUND(1,-6,2):: CAL
L PATTERN(#3,132):: CALL WA(
30):: NEXT B :: CALL POSITIO
```



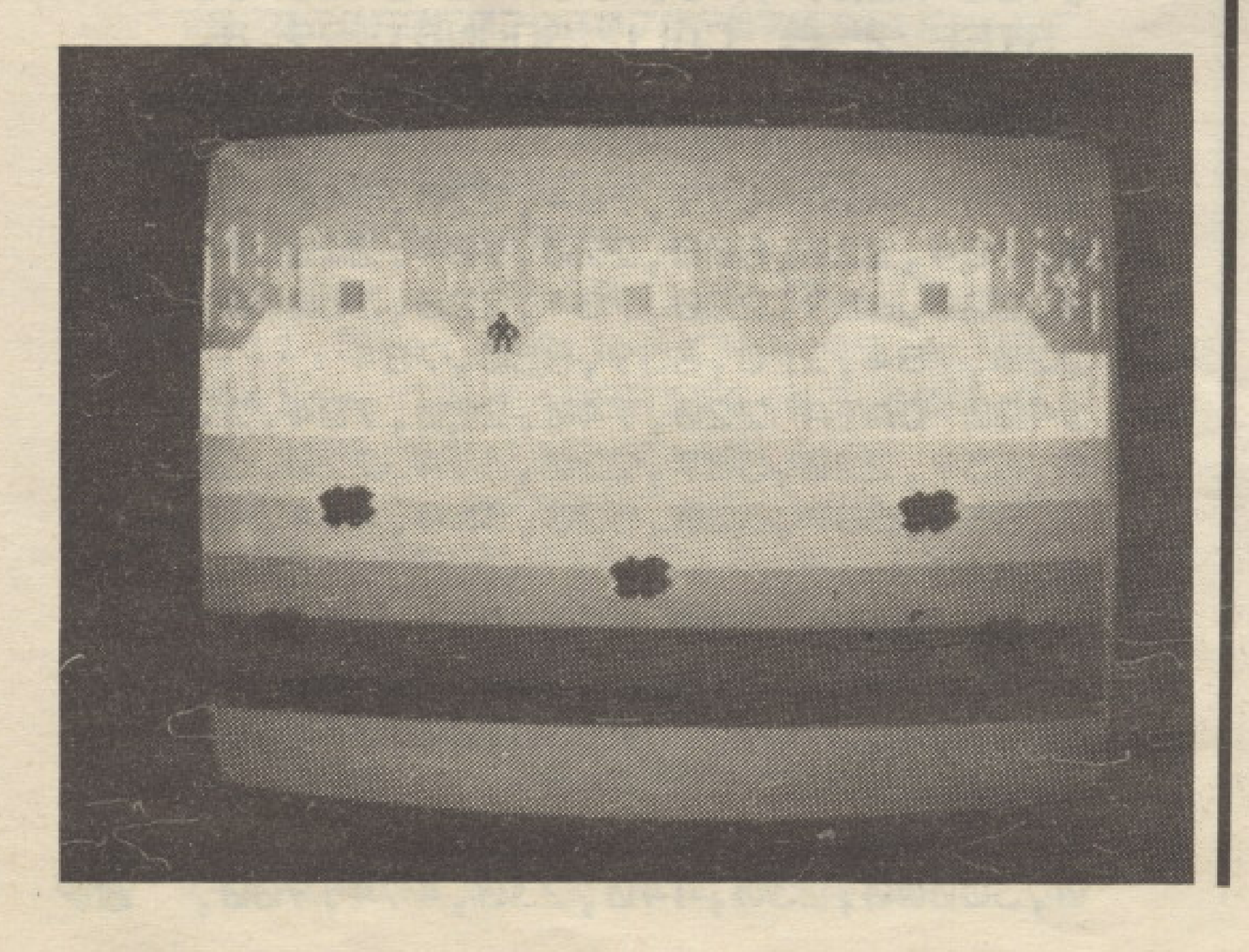
LISTINGS

```
N(#A,X1,Y1):: CALL LOCATE(#1
,X1,Y1,#3,X1,Y1):: CALL WA(I
NT(RND*1500)+300):: GOTO 390
800 TOL=TOL+1 :: IF TOL>1 TH
EN 370 ELSE CALL HCHAR(1,A,6
1):: CALL HCHAR(1,A+1,62)
810 CALL PATTERN(#1,88):: CA
LL GJ(2,110,210,50)
820 CALL PA(2):: CALL SOUND(
1,-5,0):: CALL PATTERN(#1,10
8):: CALL POSITION(#1,X1,Y1)
:: CALL SPRITE(#2,120,11,X1-
8,Y1)
830 CALL SOUND(1,-7,0):: LOC
=INT(RND*5)+5 :: CALL POSITI
ON(#LOC,X1,Y1):: CALL SPRITE
(#3,116,1,X1,Y1):: FOR A=1 T
O INT(RND*15)+7
840 CALL COLOR(#2,14,#LOC,7,
#3,10):: CALL PATTERN(#3,116
):: CALL SOUND(-999,-5,A+7):
: CALL COLOR(#2,11,#LOC,9,#3
,7):: CALL PATTERN(#3,76)
850 CALL SOUND(-999,-6,A+7):
: CALL COLOR(#2,16,#LOC,10,#
3,9):: CALL PATTERN(#3,84)::
 CALL SOUND(-333,-7,A+7):: N
EXT A
860 CALL COLOR(#LOC,2):: CAL
L DELSPRITE(#2,#3):: CALL PA
TTERN(#1,88):: CALL GJ(1,900
,925,25):: CALL PA(1):: CALL
 GJ(1,800,850,25):: GOTO 230
870 CALL GJ(2,622,644,2)
880 TE1=0 :: CALL SCREEN(2):
: FOR A=1 TO 14 :: CALL COLO
R(A,1,1):: NEXT A :: CALL CL
EAR :: CALL DELSPRITE(ALL)::
 CALL MAGNIFY(4)
890 CALL CHAR(100,"010101030
3070F0F07070F1F1F3F7FFF0080C
0E0C0C0E0E0E0F0F0F0F0F8FEFF"
,104,"0001030703030707070F0F07
03070F1F3FC0E0F0F8F8F8F0E0F0
F8F8F0E0F0FCFE")
900 CALL CHAR(108,"7F1F1F0F0
703030F0F0703070F070301FFFEF
EF4E0F8F0F0E0F0E0F0F8F8F0C0"
,112,"7F3F1F1F1F0F0707070707
0303030301FFFEFEEEECCE84C0E0
E0E0C0C0808000")
910 CALL COLOR(1,2,2,6,5,5,1
2,11,5):: CALL HCHAR(9,1,73,
256):: CALL SPRITE(#3,108,2,
65,195,#4,112,2,65,68)
920 CALL SPRITE(#2,108,2,41,
1,#7,100,2,119,211,#8,112,2,
41,134)
930 CALL SPRITE(#5,100,2,98,
130,#6,104,2,98,10,#1,92,2,1
01,230):: CALL HCHAR(12,3,12
0):: CALL HCHAR(13,3,121)
940 CALL GJ(1,196,198,2):: C
ALL GJ(1,208,210,2):: CALL G
J(1,220,222,2):: CALL GJ(1,2
33,235,2)
950 CALL MOTION(#3,0,3,#2,0,
3,#5,0,3,#4,0,3,#7,0,3,#6,0,
3,#8,0,3)
960 CALL PATTERN(#1,92):: CA
LL COLOR(12,11,5):: CALL JOY
ST(1,X,Y):: CALL MOTION(#1,-
Y*.5,X*.5):: CALL COINC(ALL,
Z):: IF Z THEN 1030
970 CALL COLOR(12,16,5):: CA
LL POSITION(#1,X1,Y1):: IF X
1<=64 THEN 1010 ELSE IF X1>=
116 THEN 1020 ELSE IF Y1>=24
8 THEN 1030
980 CALL PATTERN(#1,96):: CA
LL JOYST(1,X,Y):: CALL MOTIO
N(#1,-Y*.5,X*.5):: CALL COIN
C(ALL,Z):: IF Z THEN 1030 EL
SE CALL COLOR(12,7,5)
990 CALL COINC(#1,88,17,6,C)
:: IF C THEN 1050 ELSE 960
1000 GOTO 960
1010 CALL MOTION(#1,6,0):: C
ALL COLOR(#1,11):: TE1=TE1+1
 :: IF TE1>=4 THEN 1030 ELSE
 CALL SOUND(5,-5,0):: CALL C
OLOR(#1,2):: GOTO 960
1020 CALL MOTION(#1,-6,0)::
CALL COLOR(#1,11):: TE1=TE1+
1 :: IF TE1>=4 THEN 1030 ELS
E CALL SOUND(5,-6,0):: CALL
COLOR(#1,2):: GOTO 960
1030 FOR A=1 TO 8 :: CALL MO
TION(#A,0,0):: NEXT A :: FOR
 A=0 TO 30 STEP 2.5 :: CALL
COLOR(#1,11):: CALL SOUND(-9
9,-6,A):: CALL COLOR(#1,2)::
 CALL SOUND(-99,-7,A)
1040 NEXT A :: CALL PATTERN(
#1,128):: CALL MOTION(#1,0,2
):: CALL SO(3):: GOTO 100
1050 CALL SOUND(99,-5,0):: F
OR A=1 TO 9 :: CALL MOTION(#
A,0,0):: NEXT A :: CALL SO(4
):: CALL CLEAR :: CALL DELSP
RITE(ALL)
1060 CALL PA(1)
1070 DATA 32,,33,10101808080
8181,34,0808181010180808101 8
083820301808,36,081810101010
180808080808,38,000000000808
080881E7BD8181E7BD81
1080 DATA 40,,41,10101808080
8181,42,08081810101808081018
083820301808,44,081810101010
180808080808,46,000000000808
080881E7BD8181E7BD81
1090 DATA 48,FFFFFFFFFFFFFFF
F00000000000001E3F7F7F7F3F1
F7FFF3FFFFFFF7F7F1F3F,52,000
0000000003C7F7F7F7F1F0707000
```

```
0FFFFFFFFFFFF7B03
1100 DATA 56,FFFFFF9F1F3F1F0
F07070703070F0703FFFFFFFDFCF
8F8F8F0E0F0F0E0E0F0C,60,0307
070F0F07070F1F1F0F0F1F3F3F7F
C0E0F0F0E0E0F0F8F8F0F0F8F8F8
FCFE
1110 DATA 64,FFDFDF9F1F1F1F0
F0F07070707030303FFFDFDF8F8F
8F8F0F0F0F0F0E0E0E0C,68,0307
0F0F1F07070F0F1F3F1F0F3F7F7F
C0E0F0F0E0C0E0F0E0C0E0F0F8F0
F8FC
1120 DATA 72,FFFFFFFFFFFFFF
FFFFF78382,74,FFFFFE7C38381,
75,FFFF7E3C1E1C1,76,FFF7C3,7
7,,78,FFFF3C1,79,FFFF7F3602
1130 DATA 80,7E3C181818183C7
E0000000000000081C,84,,85,000
0000000183C18,86,,87,
1140 DATA 116,001818148F5E2E
07031F12111020000000000000000
804020000000008040204,120,00
00000000010204000000000010204
0200181828417A74E0C0F8488808
04
1150 DATA 124,000001050D0E0B
0B090B0B020202020400000080A0B0
70D0D090D0D0404040402
1160 RESTORE 1070 :: FOR A=1
 TO 31 :: READ B,B$ :: CALL
CHAR(B,B$):: NEXT A :: FOR A
=0 TO 14 :: CALL COLOR(A,1,1
):: NEXT A
1170 CALL SPRITE(#1,88,16,12
0,120,-3,0)
1180 RUN "CS1"
1190 SUB PA(X):: ON X GOTO 1
200,1230
1200 CALL CHAR(92,"003030281
C1E3DEE0C14244424122101",94,
"",95,"",96,"00606050387C7A5
959B8A8454220204",98,"",99,"
")
1210 CALL CHAR(100,"000C0C14
3878BC77302824222448848",102
,"",103,"",104,"0006060A1C3E
5E9A9A1D15A242040402",106,""
,107,"")
1220 CALL CHAR(108,"303472D9
FE787830784C464C5850402",110
,"",111,"",112,"0C2C4E9B7F1E
1E0C1E3262321A0A0204",114,""
,115,""):: SUBEXIT
1230 CALL CHAR(92,"0C0C0B770
F18F",93,"",94,"0001E5F5B90D
0702",95,"",96,"C0C3BF6F3618
0F",97,"",98,"0000FC82C07E01
",99,"")
1240 CALL CHAR(100,"0080A7AF
9DB0E04",101,"",102,"3030D0E
EF0180F",104,"00003F41037E8"
,105,"",106,"03C3FDF66C18F")
1250 CALL CHAR(108,"18245A99
DB667E3C183C7EE781422442",11
2,"42242424243C3C183C7EFFA59
9999942"):: SUBEND
1260 SUB WA(X):: FOR WA=1 TO
 X :: NEXT WA :: SUBEND
1270 SUB MU :: CALL POSITION
(#1,X1,Y1):: IF X1<=79 THEN
SUBEXIT
1280 A=INT(RND*2)+1 :: IF A=
1 THEN UM=INT(RND*3)+10 ELSE
 SUBEXIT
1290 CALL POSITION(#UM,E,R):
: IF E=179 THEN 1310
1300 CALL LOCATE(#UM,179,R):
: CALL SOUND(1,-6,19):: SUBE
XIT
1310 A=INT(RND*2)+1 :: IF A=
1 THEN CALL LOCATE(#UM,168,R
)
1320 CALL SOUND(1,-7,19):: S
UBEND
1330 SUB GJ(X,X1,X2,X3)
1340 FOR Z=1 TO X :: FOR Z1=
5 TO 30 STEP 5 :: FOR Z2=X1
TO X2 STEP X3
1350 CALL SOUND(1,Z2,Z1,Z2+2
00,Z1,Z2*Z,Z1-2):: NEXT Z2 :
: NEXT Z1 :: NEXT Z :: SUBEN
D
1360 SUB SO(X):: ON X GOTO 1
370,1410,1480,1510
1370 DATA 466,988,370,740,34
9,698,311,622,294,587,247,49
4,233,466,247,494,294,587,31
1,622
1380 DATA 294,587,247,494,23
3,466,208,415,185,370,175,34
9,156,311,156,311,156,311
1390 RESTORE 1370 :: FOR A=1
 TO 19 :: READ B,C :: FOR D=
0 TO 16 STEP 8 :: CALL SOUND
(-99,B,2,C,D,C+2,D):: CALL S
OUND(-399,B,D,B+2,D,C,2):: N
EXT D
1400 NEXT A :: FOR A=2 TO 30
 STEP 2 :: CALL SOUND(-99,B,
A,C,A,C+2,A):: NEXT A :: SUB
EXIT
1410 DATA 999,784,220,740,22
0,784,220,880,220,784,220,74
0,220,659,220,784,220,30000,
220,784,220,659,850,784
1420 DATA 220,740,220,784,22
0,659,220,587,220,494,220,52
3,850,587,350,523,350,494,45
0,440
1430 DATA 230,494,230,30000,
230,494,230,440,700,494,230,
440,230,494,220,30000,230,49
4,230,523,750,440
1440 DATA 230,392,230,440,21
0,30000,230,440,230,494,900,
```

LISTINGS

```
392
1450 RESTORE 1410 :: FOR A=1
 TO 39 :: READ B,C :: CALL S
OUND(B,C,5,C/2,6,C/2+2,6)::
CALL KEY(1,K,S):: IF K=18 TH
EN 1460 ELSE 1470
1460 FOR A=6 TO 30 STEP 2 ::
 CALL SOUND(-99,C,A,C/2,A,C/
2+2,A):: NEXT A :: SUBEXIT
1470 NEXT A :: GOTO 1450
1480 FOR D=1 TO 3 :: RESTORE
 1370 :: FOR A=1 TO 19 :: RE
AD B,C
1490 CALL POSITION(#1,X1,Y1)
:: IF Y1>=250 THEN CALL DELS
PRITE(#1)
1500 CALL SOUND(225,B,30,C,3
0,C*2,30,-4,5):: NEXT A :: N
EXT D :: SUBEXIT
1510 DATA 311,370,415,466,55
4,554,554,554,554,466,415,37
0,311,311,311,311
1520 DATA 311,370,415,466,55
4,554,554,554,554,466,415,37
0,311,311,311,311
1530 DATA 370,311,277,233,27
7,277,277,277,277,311,370,41
5,370,370,370,370
1540 DATA 370,311,277,233,27
7,277,277,277,277,311,370,41
5,370,370,370,370
1550 DATA 311,370,415,466,55
4,554,554,554,554,466,415,37
0,311,311,311,311
1560 DATA 311,370,415,466,55
4,554,554,554,554,466,415,37
0,311,277,311,277,311,277,31
1,277,311,311,311,311,311
1570 RESTORE 1510 :: FOR A=1
 TO 105 :: READ B :: FOR C=0
 TO 30 STEP 15 :: CALL SOUND
(-333,B,5,B,C,B,C)
1580 NEXT C :: NEXT A :: SUB
END
```



LISTINGS

```
10 ! **********************
11 ! * DAS GEHEIMNIS  VON *
12 ! *    CHICHEN ITZA    *
13 ! *      (Teil 3)      *
14 ! *                    *
15 ! *    Copyright by    *
16 ! *                    *
17 ! *  Manfred Lipowski  *
18 ! *                    *
19 ! * Benoetigte Geraete *
20 ! *  TI99/4A  Konsole  *
21 ! *     Ext. Basic     *
22 ! *     Joystick 1     *
23 ! *    Cassettenrec.   *
24 ! *(oder DISK+32K-Erw.)*
25 ! *                    *
26 ! *  Speicherbelegung  *
27 ! *    12439 Bytes     *
28 ! *                    *
29 ! **********************
100 CALL CLEAR :: CALL MAGNI
FY(3):: CALL DELSPRITE(ALL):
: RANDOMIZE :: CALL SCREEN(2
):: CALL TO1(-1):: BM=0
110 CALL HCHAR(1,1,72,64)::
CALL HCHAR(3,1,73):: CALL HC
HAR(3,2,74):: CALL HCHAR(3,3
,76):: FOR A=26 TO 32 :: B=I
NT(RND*4)+73
120 CALL HCHAR(3,A,B):: NEXT
 A :: CALL HCHAR(4,1,77,64):
: FOR A=7 TO 11 :: FOR B=1 T
O 32
130 C=INT(RND*7)+32 :: CALL
HCHAR(A,B,C):: NEXT B :: NEX
T A :: FOR A=12 TO 16 :: FOR
 B=1 TO 32 :: C=INT(RND*7)+4
0 :: CALL HCHAR(A,B,C):: NEX
T B
140 NEXT A :: CALL TO1(-2)::
 FOR A=17 TO 21 :: FOR B=1 T
O 32 :: C=INT(RND*7)+32 :: C
ALL HCHAR(A,B,C):: NEXT B ::
 NEXT A
150 CALL HCHAR(22,1,72,96)::
 CALL VCHAR(7,3,39,5):: CALL
 VCHAR(12,30,47,5):: CALL VC
HAR(17,3,39,5):: CALL HCHAR(
6,1,77,2)
160 CALL HCHAR(6,31,81,2)::
FOR A=1 TO 3 :: CALL HCHAR(2
1+A,30,48,3):: NEXT A :: CAL
L HCHAR(22,29,51)
170 CALL HCHAR(23,28,52):: C
ALL HCHAR(23,29,50):: CALL H
CHAR(24,28,53):: CALL HCHAR(
24,29,54)
180 DISPLAY AT(3,2)SIZE(22):
"8:NOIMMIKJ@BKNIMMION@:" ::
DISPLAY AT(4,2)SIZE(22):"9;M
MMMMMMMACMMMMMMMM9C"
190 DISPLAY AT(5,2)SIZE(22):
"<>MMMMMMMMDFMMMMMMMM<F" ::
```

```
DISPLAY AT(6,1):"M=?MMQMMMMQ
EGMQMMMMMQ=GMQQMP"
200 CALL COLOR(1,7,11,2,13,4
,3,5,7,4,7,2,5,7,2,6,7,2,7,1
3,2)
210 B=1 :: CALL SPRITE(#1,68
,7,33,64,0,0,#2,60,7,33,144,
0,0,#3,56,7,17,64,0,0,#4,64,
7,17,144,0,0)
220 CALL CHAR(140,RPT$("F",6
4)):: CALL SPRITE(#5,84,16,2
6,233,0,0,#6,140,5,184,232,0
,0,#7,92,2,190,232,0,0)
230 CALL P(7,92):: CALL POSI
TION(#7,X1,Y1):: IF X1=175 T
HEN 250 ELSE CALL WA(40):: C
ALL P(7,96)
240 CALL LOCATE(#7,X1-B,Y1):
: CALL S(1,6,9):: CALL WA(40
):: GOTO 230
250 CALL S(1,5,0):: CALL P(7
,116):: FOR A=-6 TO 4 STEP 2
 :: CALL MOTION(#7,A,-3):: C
ALL WA(99):: NEXT A :: CALL
MOTION(#7,0,0)
260 CALL S(1,7,0):: CALL P(7
,108):: CALL WA(35):: CALL S
(1,6,2):: CALL P(7,112)
270 CALL LOCATE(#7,169,208):
: CALL S(1,5,0):: CALL WA(30
):: CALL P(7,92):: CALL S(1,
7,0):: CALL LOCATE(#7,169,20
0)
280 CALL P(7,96):: CALL WA(3
0):: CALL S(1,7,9):: CALL P(
7,88):: CALL POSITION(#7,X1,
Y1)
290 CALL DELSPRITE(#6):: CAL
L SPRITE(#6,88,2,X1,Y1):: CA
LL DELSPRITE(#7):: FOR A=1 T
O 3 :: CALL P(6,88)
300 CALL SOUND(150,196,0)::
CALL SOUND(200,262,0):: CALL
 P(6,124):: CALL SOUND(400,1
96,3,262,3,330,1):: CALL WA(
150):: NEXT A
310 CALL P(6,88):: CALL PA(1
):: PAT=92 :: CALL TO1(-3)::
 CALL SPRITE(#7,128,4,169,12
0,0,0,#8,136,4,169,104,0,0):
: CALL TO1(-6):: TE1=0
320 CALL PATTERN(#6,88):: CA
LL COLOR(#5,5):: CALL COINC(
ALL,Z):: IF Z THEN 930
330 CALL MO(Z):: IF Z THEN 9
30 ELSE CALL JOYST(1,X,Y)::
CALL POSITION(#6,X1,Y1):: CA
LL COLOR(#5,16)
340 IF X=-4 THEN PAT=92 :: Y
1=Y1-8 :: GOTO 370 ELSE IF X
=4 THEN PAT=100 :: Y1=Y1+8 :
: GOTO 370
350 CALL COINC(ALL,Z):: IF Z
 THEN 930
360 IF Y=-4 THEN X1=X1+8 ::
GOTO 370 ELSE IF Y=4 THEN X1
=X1-8 :: GOTO 370 ELSE 320
370 IF Y1<8 THEN 420 ELSE IF
 X1>=184 THEN 320 ELSE IF Y1
>209 THEN 320
380 CALL PATTERN(#6,PAT):: C
ALL S(1,5,3):: CALL GCHAR(X1
/8+2,Y1/8+1,AS):: IF AS<>72
THEN 390 ELSE 410
390 IF AS=39 THEN 430 ELSE 1
130
400 CALL COINC(ALL,Z):: IF Z
 THEN 930
410 CALL COLOR(#5,5):: CALL
S(1,6,0):: CALL LOCATE(#6,X1
,Y1):: CALL PATTERN(#6,PAT+4
):: GOTO 330
420 CALL P(6,88):: TE1=TE1+1
 :: IF TE1=2 THEN 1130 ELSE
CALL TO1(-5):: CALL LOCATE(#
6,169,200):: CALL TO1(-7)::
GOTO 320
430 CALL LE(6):: CALL TO1(-5
):: CALL DELSPRITE(#7,#8)::
CALL TO1(-6):: CALL PA(2)::
CALL SPRITE(#7,136,2,115,248
,5,-7,#8,136,2,92,110,-5,-7)
440 TE1=0
450 CALL PATTERN(#6,88,#7,13
6,#8,140):: CALL COLOR(#5,14
):: CALL COINC(ALL,Z):: IF Z
 THEN 930
460 CALL JOYST(1,X,Y):: CALL
 POSITION(#6,X1,Y1,#7,X2,Y2)
:: IF X2>114 THEN CALL MOTIO
N(#7,-6,-6,#8,6,-6)ELSE IF X
2<91 THEN CALL MOTION(#7,6,-
6,#8,-6,-6)
470 CALL COLOR(#5,16):: CALL
 COINC(ALL,Z):: CALL PATTERN
(#7,140,#8,136):: IF Z THEN
930
480 IF X=-4 THEN PAT=92 :: Y
1=Y1-8 ELSE IF X=4 THEN PAT=
100 :: Y1=Y1+8 ELSE IF X=0 A
ND Y=0 THEN 450
490 IF Y1<8 THEN 540 ELSE IF
 Y1>248 THEN 530 ELSE CALL C
OLOR(#5,14)
500 CALL S(1,5,3):: CALL PAT
TERN(#6,PAT):: CALL GCHAR(X1
/8+2,Y1/8+1,PP):: IF PP=47 T
HEN 550
510 CALL COINC(ALL,Z):: IF Z
 THEN 930
520 CALL PATTERN(#7,136,#8,1
40):: CALL S(1,6,9):: CALL L
OCATE(#6,X1,Y1):: CALL PATTE
RN(#6,PAT+4):: GOTO 460
530 CALL P(6,88):: CALL TO1(
-7):: CALL LOCATE(#6,X1,16):
```

```
: CALL TO1(-6):: GOTO 450
540 TE1=TE1+1 :: IF TE1=2 TH
EN 1130 ELSE CALL SOUND(99,3
49,0):: CALL P(6,88):: CALL
SOUND(99,294,0):: CALL LOCAT
E(#6,X1,16):: GOTO 450
550 CALL P(6,88):: CALL LOCA
TE(#6,X1,Y1):: CALL TO1(-3):
: CALL DELSPRITE(#7,#8):: CA
LL TO1(-1):: CALL LE(5):: CA
LL PA(3):: TE1=0
560 CALL SPRITE(#7,128,2,73,
32,0,3,#8,128,2,73,128,0,3,#
9,136,2,56,5,0,6)
570 CALL COLOR(#5,8):: CALL
PATTERN(#6,88,#7,128,#8,132,
#9,136):: CALL S(1,5,0):: CA
LL COINC(ALL,Z):: IF Z THEN
930
580 CALL JOYST(1,X,Y):: CALL
 POSITION(#6,X1,Y1):: CALL C
OLOR(#5,11):: J=INT(RND*7)
590 CALL COINC(ALL,Z):: IF Z
 THEN 930 ELSE CALL PATTERN(
#7,132,#8,128,#9,140)
600 CALL KEY(1,K,S):: IF K=1
8 THEN 670
610 CALL MOTION(#9,0,6+J)::
CALL COLOR(#5,16)
620 IF X=-4 THEN PAT=92 :: Y
1=Y1-8 ELSE IF X=4 THEN PAT=
100 :: Y1=Y1+8 ELSE IF X=0 A
ND Y=0 THEN 570
630 IF Y1<8 THEN Y1=248 ELSE
 IF Y1>248 THEN 720
640 CALL COLOR(#5,8):: CALL
PATTERN(#6,PAT):: CALL S(1,6
,0):: CALL GCHAR(X1/8+2,Y1/8
+1,PO):: IF PO=39 THEN 730
650 CALL PATTERN(#7,128,#8,1
32,#9,136):: CALL S(1,7,0)::
 CALL LOCATE(#6,X1,Y1):: CAL
L PATTERN(#6,PAT+4):: CALL C
OINC(ALL,Z)
660 IF Z THEN 930 ELSE 580
670 IF Y1<32 THEN CALL S(-33
3,3,0):: GOTO 580
680 IF X=-4 THEN PA1=116 ::
BR=-8 :: GOTO 690 ELSE IF X=
4 THEN PA1=120 :: BR=8 :: GO
TO 690 ELSE CALL S(-300,1,9)
:: GOTO 580
690 CALL PATTERN(#6,PA1):: C
ALL S(9,6,0):: FOR A=-8 TO 8
 :: CALL MOTION(#6,A,BR):: C
ALL COINC(ALL,Z):: IF Z THEN
 930 ELSE CALL POSITION(#6,D
T,TD)
700 NEXT A :: CALL MOTION(#6
,0,0)
710 CALL LOCATE(#6,X1,TD)::
GOTO 570
720 TE1=TE1+1 :: IF TE1=2 TH
EN 1130 ELSE CALL SOUND(99,2
08,0):: CALL SOUND(99,311,0)
:: CALL LOCATE(#6,X1,Y1-16):
: GOTO 570
730 CALL LOCATE(#6,X1,16)::
CALL P(6,88):: CALL S(9,5,0)
740 CALL TO1(-6):: CALL DELS
PRITE(#7,#8,#9):: CALL TO1(-
5):: CALL LE(5):: FOR A=1 TO
 3
750 CALL SOUND(175,196,0)::
CALL SOUND(200,262,0):: CALL
 SOUND(225,330,0,262,2):: CA
LL SOUND(250,392,0,330,2)
760 CALL SOUND(450,523,0,392
,2,330,4):: NEXT A :: TE,TE1
=0
770 CALL POSITION(#6,X1,Y1):
: CALL LOCATE(#6,X1+1,Y1)
780 CALL P(6,88):: CALL COLO
R(#5,16):: CALL COLOR(7,4,2)
790 CALL JOYST(1,X,Y):: CALL
 POSITION(#6,X1,Y1)
800 CALL KEY(1,K,S):: IF K=1
8 THEN 850 ELSE CALL COLOR(#
5,11):: CALL COLOR(7,13,2)::
 CALL COLOR(#5,8)
810 IF X=-4 THEN PAT=92 :: Y
1=Y1-8 ELSE IF X=4 THEN PAT=
100 :: Y1=Y1+8 ELSE IF X=0 A
ND Y=0 THEN 780
820 IF Y1<8 THEN 920 ELSE CA
LL PATTERN(#6,PAT):: CALL S(
1,5,0):: CALL GCHAR(X1/8+2,Y
1/8+1,PL):: IF PL=81 THEN 90
0 ELSE IF PL=80 THEN 950
830 CALL COLOR(#5,16):: CALL
 COLOR(7,4,2)
840 CALL S(1,5,6):: CALL LOC
ATE(#6,X1,Y1):: CALL PATTERN
(#6,PAT+4):: CALL WA(15):: G
OTO 790
850 IF Y1<32 THEN CALL SOUND
(-333,-3,0):: GOTO 780 ELSE
TE=TE+1 :: IF TE<=7 THEN 860
 ELSE 780
860 IF X=-4 THEN PA1=116 ::
BR=-8 :: GOTO 870 ELSE IF X=
4 THEN PA1=120 :: BR=8 :: GO
TO 870 ELSE CALL S(-300,5,4)
:: GOTO 780
870 CALL PATTERN(#6,PA1):: C
ALL S(9,6,0):: FOR A=-6 TO 6
 :: CALL MOTION(#6,A,BR):: C
ALL SOUND(-99,-5,15+A)
880 NEXT A :: CALL MOTION(#6
,0,0):: CALL S(1,5,0):: CALL
 POSITION(#6,TD,Y1):: CALL L
OCATE(#6,X1,Y1+BR)
890 CALL GCHAR(X1/8+2,Y1/8+1
,PK):: IF PK=81 THEN 900 ELS
E 780
900 CALL LOCATE(#6,X1,Y1)::
```

LISTINGS

```
FOR A=1 TO 8 :: CALL COLOR(7
,4,2):: CALL S(-333,5,A*2)::
 CALL COLOR(#6,8):: CALL COL
OR(7,3,2):: CALL S(-333,6,A*
2)
910 CALL COLOR(#6,11):: CALL
 COLOR(7,13,2):: CALL S(-333
,7,A*2):: CALL COLOR(#6,2)::
 NEXT A :: BM=1 :: GOTO 1130
920 TE1=TE1+1 :: IF TE1=2 TH
EN 1130 ELSE CALL SOUND(99,2
94,0):: CALL SOUND(99,247,0)
:: CALL LOCATE(#6,X1,Y1+16):
: CALL SOUND(99,196,0):: GOT
O 780
930 CALL MOTION(#6,0,0,#7,0,
0,#8,0,0,#9,0,0):: FOR A=1 T
O 9 :: CALL S(-99,5,A*2):: C
ALL COLOR(#6,16):: CALL S(-9
9,6,A*2):: CALL COLOR(#6,14)
940 CALL S(-99,7,A*2):: CALL
 COLOR(#6,2):: NEXT A :: GOT
O 1130
950 CALL LOCATE(#6,X1,Y1)::
FOR A=200 TO 2500 STEP 25 ::
 CALL SOUND(-999,A,30,A,30,A
,30,-4,0):: NEXT A
960 FOR A=0 TO 25 :: CALL SO
UND(-99,-5,A,220,A+5):: CALL
 SOUND(-99,-7,A,110,A+5):: N
EXT A :: CALL TO1(-5)
970 CALL CLEAR :: CALL DELSP
RITE(ALL):: CALL CHARSET ::
FOR A=1 TO 8 :: CALL COLOR(A
,7,1):: NEXT A :: CALL SCREE
N(16)
980 CALL COLOR(0,14,1,9,5,1,
10,5,1,11,5,1,12,13,13):: DI
SPLAY AT(2,1):"BRAVO DR.";NA
M$;" !": :"SIE HABEN ES GESC
HAFFT !"
990 CALL CHAR(96,"1F3F6F6360
514020202015112EE0E0F1FF0080C0
C0C040408080000080E0F8FCFE",
100,"7FFFFFFFFFFFFFFF7F7F7F7
FFFFFFFFFDEEFEFEFEFFFEFEECEC
ECECEEEE6EEEA")
1000 CALL CHAR(104,"F1F1F1E0
E0E0E0E0E0C0C0C0C040C0E2EC
F0F0F0F0F8F878787878645E7F
",108,RPT$("0",27)&"3070F0F1
E1E1E1E1F1E0E0E0E0E0E0C0E0
A")
1010 CALL CHAR(112,"08060101
01010101030303030303040F1F",
120,RPT$("F",16),128,"000070
FCFEFF7F3F",130,"00000E3F7FF
FFEFC",129,"",131,"")
1020 CALL CHAR(116,"03040A08
05344A4B4A281E03",118,"C0205
010A02C52D2521478C",124,"030
70F0F07377F7F7F3F1F03",126,"
C0E0F0F0E0ECFEFEFEFCF8C")
1030 CALL MAGNIFY(4):: CALL
HCHAR(24,11,120,11):: CALL S
PRITE(#3,128,9,171,112,#2,12
4,11,157,112,#1,116,2,157,11
2)
1040 DATA " `b","mac","ndf",
"oeg","phj","qik"
1050 RESTORE 1040 :: FOR A=1
 TO 6 :: READ A$ :: DISPLAY
AT(13+A,13):A$ :: NEXT A
1060 DISPLAY AT(6,1):"DER SC
HATZ GEHOERT IHNEN." :: FOR
A=1 TO 3
1070 CALL SOUND(175,196,0)::
 CALL SOUND(200,262,0):: CAL
L SOUND(225,330,0,262,2):: C
ALL SOUND(250,392,0,330,2)
1080 CALL SOUND(450,523,0,39
2,2,330,4):: NEXT A :: CALL
WA(99):: FOR A=0 TO 25 :: CA
LL COLOR(#2,14):: CALL SOUND
(-333,-5,A):: CALL COLOR(#2,
11):: NEXT A
1090 FOR A=30 TO 0 STEP -1 :
: CALL COLOR(#2,14):: CALL S
OUND(-333,-5,A):: CALL COLOR
(#2,11):: NEXT A
1100 FOR A=0 TO 30 :: CALL C
OLOR(9,7,1,10,7,1,11,7,1)::
CALL SOUND(-99,-1,A):: CALL
COLOR(9,3,1,10,3,1,11,3,1)::
 CALL SOUND(-99,-2,A)
1110 CALL COLOR(9,5,1,10,5,1
,11,5,1):: CALL SOUND(-99,-3
,A):: NEXT A
1120 CALL TO1(-1):: CALL TO1
(-7):: A=8 :: DISPLAY AT(10,
1):"DAS IST HIER DIE FRAGE !
" :: GOTO 1210
1130 CALL TO1(-7):: CALL DEL
SPRITE(#6,#7,#8,#9):: CALL K
EY(1,K,S):: CALL TO1(-6):: I
F K=18 THEN 1140 ELSE IF BM<
>1 THEN 220 ELSE CALL TO1(-2
)
1140 CALL TO1(-6):: CALL CLE
AR :: CALL DELSPRITE(ALL)::
CALL CHARSET :: CALL CHAR(96
,RPT$("FF",8)):: CALL COLOR(
0,13,1,9,5,5):: FOR A=1 TO 8
1150 CALL COLOR(A,16,1):: NE
XT A :: CALL PA(4):: CALL SC
REEN(2):: CALL VCHAR(1,26,96
,168):: FOR A=6 TO 13
1160 CALL SPRITE(#A,140,INT(
RND*2)+15,INT(RND*180)+10,IN
T(RND*42)+202,INT(RND*30)+6,
0):: NEXT A
1170 CALL SPRITE(#1,140,15,3
0,220,9,0,#2,140,16,99,228,9
,0,#3,140,15,173,224,9,0)
1180 CALL SPRITE(#4,124,2,19
9,224,-3,0,#5,136,11,199,224
```

```
,-3,0)
1190 DISPLAY AT(2,1)SIZE(5):
"SORRY" :: DISPLAY AT(4,1)SI
ZE(21):"DR:";NAM$;" !" :: DI
SPLAY AT(6,1)SIZE(13):"SIE H
ABEN DAS"
1200 DISPLAY AT(8,1)SIZE(16)
:"GEHEIMNIS LEIDER" :: DISPL
AY AT(10,1)SIZE(15):"NICHT G
ELOEST !" :: A=12
1210 DISPLAY AT(A,1)SIZE(15)
:"NOCHMAL J-N ? J" :: ACCEPT
 AT(A,15)SIZE(-1)BEEP VALIDA
TE("JN"):J$
1220 CALL DELSPRITE(ALL):: I
F J$="J" THEN 1230 ELSE CALL
 CLEAR :: STOP
1230 RUN "CS1"
1240 SUB PA(X):: ON X GOTO 1
250,1280,1300,1330
1250 CALL CHAR(128,"00000000
4060F8FDFFCFB7BBDDEE07050000
00000000F0BEFFEAC0D5FF7E0080
",132,"0000000020B0F8FDFFCFB
7B7B570E0B000000000000000F0DEF
FEAD5FFFE000000")
1260 CALL CHAR(136,"00000000
00183163CFFF7F1F030100000000
000092DBFFFFFF9F6F77BBDD0E0B
",140,"000000000000001D336FFF7
F1F020103020000000000496DFFFFF
F9F6F6FEFDF80C0")
1270 SUBEXIT
1280 CALL CHAR(136,"0C1F1F0F
0F0763F63FEF070100010000C060
A0A0C0C060F0E0C0A02090200000
",140,"0000000000000367EC3FF
70703010000000000000000E0F07
0E0C0E8E8F4F468")
1290 SUBEXIT
1300 CALL CHAR(128,"00000000
000000000000000080C0476D380000
00000000000000000000000061FB6E0
",132,"000000000000000000000
0000E1BB1610000000000000000000
00000061F3660C0")
1310 CALL CHAR(136,"03060505
0303060F07030504090400000030F8
F8F0F0E0C66FFCF7E08000800000
",140,"00000000000000070F0E070
317172F2F160000000000000C0E63
7FCEFE0C0800000")
1320 SUBEXIT
1330 CALL CHAR(136,"387C7C7F
7F7F7F3F3F3F3F3F1F1E0E041C3E
3EFEFEFEFEFEFCFCFCFCFCF8787020
",140,"1E7FFFFFFFFFFFF7F3F1F0
30000000000000098FCFEFFFFFFFFEF
CF8C000000000000")
1340 SUBEND
1350 SUB WA(X):: FOR Z=1 TO
X :: NEXT Z :: SUBEND
1360 SUB MO(Z)
1370 CALL POSITION(#7,A1,A2,
#8,A3,A4):: CALL SOUND(1,-7,
0):: CALL PATTERN(#7,132,#8,
140):: PX=INT(RND*2):: IF PX
 THEN 1380 ELSE 1410
1380 CALL COINC(ALL,Z):: PX=
INT(RND*2):: IF PX THEN 1400
1390 IF A4<=32 THEN 1440 ELS
E A4=A4-8 :: A2=A2-8 :: GOTO
 1440
1400 IF A2>=200 THEN 1440 EL
SE A2=A2+8 :: A4=A4+8 :: GOT
O 1440
1410 CALL COINC(ALL,Z):: PX=
INT(RND*2):: IF PX THEN 1420
 ELSE 1430
1420 IF A1<=168 THEN 1440 EL
SE A1=A1-8 :: A3=A3-8 :: GOT
O 1440
1430 IF A1>176 THEN 1440 ELS
E A1=A1+8 :: A3=A3+8
1440 CALL LOCATE(#7,A1,A2,#8
,A3,A4):: CALL SOUND(1,-7,9)
:: CALL PATTERN(#7,128,#8,13
6):: CALL COINC(ALL,Z):: SUB
END
1450 SUB TO1(X):: FOR Z=30 T
O 0 STEP -2 :: CALL SOUND(-9
9,X,Z):: NEXT Z :: FOR Z=0 T
O 16 STEP 2 :: CALL SOUND(-9
9,X,Z):: NEXT Z
1460 FOR Z=15 TO 0 STEP -1 :
: CALL SOUND(-99,X,Z):: NEXT
 Z :: FOR Z=3 TO 30 STEP 3 :
: CALL SOUND(-99,X,Z):: NEXT
 Z :: SUBEND
1470 SUB LE(X):: FOR A=1 TO
X :: CALL P(6,108):: CALL PO
SITION(#6,X1,Y1):: CALL SOUN
D(1,-6,A):: CALL LOCATE(#6,X
1-8,Y1):: CALL WA(25)
1480 IF X1<55 THEN CALL COLO
R(#6,11)
1490 CALL P(6,112):: CALL SO
UND(1,-7,A):: CALL WA(25)::
NEXT A :: CALL P(6,88):: SUB
END
1500 SUB S(X,Y,Z):: CALL SOU
ND(X,-Y,Z):: SUBEND
1510 SUB P(Q,W):: CALL PATTE
RN(#Q,W):: SUBEND
```

LISTINGS

```
10 ! **********************
11 ! * INTEGRALBERECHNUNG *
12 ! * (Simpson-Naeherung)*
13 ! *                    *
14 ! *   Copyright by     *
15 ! *                    *
16 ! *   Rolf Offerhaus   *
17 ! *                    *
19 ! * Benoetigte Geraete *
20 ! *  TI99/4A  Konsole  *
21 ! *     Ext. Basic     *
22 ! *                    *
26 ! *  Speicherbelegung  *
27 ! *     3124 Bytes     *
28 ! *                    *
29 ! **********************
100 CALL CLEAR :: CALL SCREE
N(5):: FOR I=0 TO 14 :: CALL
 COLOR(I,16,5):: NEXT I :: O
N WARNING NEXT
110 CALL CHAR(91,"0F1C383030
30303",92,"303030301C0C0C0C"
,93,"0C0C0C0C0C1C38F")
120 CALL CHAR(123,"FF8080808
0808080",124,"FF010101010101
01",125,"80808080808080FF",1
26,"01010101010101FF")
130 CALL CHAR(33,"FF",35,"00
000000000000FF",36,"80808080
8080808",37,"010101010101010
1")
140 A=3 :: GOSUB 550
150 DISPLAY AT(10,1):" 1.INS
TRUKTIONEN": :" 2.STETIG": :
" 3.UNSTETIG": :" 4.PROGRAMM
ENDE" :: DISPLAY AT(20,2):"B
ITTE WAEHLEN SIE"
160 CALL KEY(0,K,S):: IF K=4
9 THEN 170 ELSE IF K=50 THEN
 250 ELSE IF K=51 THEN 380 E
LSE IF K=52 THEN CALL CLEAR
:: STOP ELSE 160
170 A=3 :: GOSUB 550 :: DISP
LAY AT(10,1):"STETIG:":"Dies
es Programm verwendet...man
fuer die naeherungsweiseBere
chnung des Integrals l"
180 DISPLAY AT(14,1):"einer
vom Anwender definier-ten Fu
nktion ueber dem......Interv
all A bis B"
190 DISPLAY AT(18,7):"[b" ::
 DISPLAY AT(19,5):"l=\ F(x)
dx" :: DISPLAY AT(20,7):"]a"
200 DISPLAY AT(24,1):"Weiter
 mit beliebiger taste" :: CA
LL KEY(0,K,S):: IF S=0 THEN
200
210 CALL HCHAR(10,1,32,480):
: DISPLAY AT(10,1):"UNSTETIG
:":"Dieses Programm verwende
t...man zur naeherungsweisen
"
220 DISPLAY AT(13,1):"Berech
nung des Integrals l ,wobei:
" :: DISPLAY AT(16,5):"l=\ F
(x) dx" :: DISPLAY AT(15,7):
"[b" :: DISPLAY AT(17,7):"]a
":"ist"
230 DISPLAY AT(19,1):"Der We
rt von F(x) muss an...n+1 gl
eich untergliederten..Punkte
n gegeben sein"
240 DISPLAY AT(24,1):"Weiter
 mit beliebiger Taste" :: CA
LL KEY(0,K,S):: IF S=0 THEN
240 ELSE 140
250 A=4 :: GOSUB 550 :: DISP
LAY AT(4,11)SIZE(6):"STETIG"
 :: DISPLAY AT(10,1):"Die Fu
nktion F(x) kann in...zeile
260 definiert werden"
260 DEF F(X)=SIN(X)
270 DISPLAY AT(14,1)SIZE(13)
:"Eingabe von A" :: ACCEPT A
T(14,15)VALIDATE(NUMERIC):A
280 DISPLAY AT(15,1)SIZE(13)
:"Eingabe von B" :: ACCEPT A
T(15,15)VALIDATE(NUMERIC):B
290 DISPLAY AT(16,1)SIZE(13)
:"Eingabe von N" :: DISPLAY
AT(17,1):"N(N=2,4,6...)" ::
ACCEPT AT(16,15)VALIDATE(NUM
ERIC):N :: IF N=0 THEN 290
300 IF N/2<>INT(N/2)THEN 290
310 ER=0 :: H=(B-A)/N
320 FOR I=2 TO N STEP 2
330 ER=ER+F(A+((I-1)*H))*4+F
(A+(I*H))*2
340 NEXT I
350 ER=ER-F(A+(N*H))*2 :: ER
=(ER+F(A)+F(B))*H/3
360 DISPLAY AT(19,1):"L=";ER
 :: DISPLAY AT(24,1):"Weiter
 mit REDO oder BACK"
370 CALL KEY(0,K,S):: IF K=1
5 THEN 140 ELSE IF K=6 THEN
CALL HCHAR(14,1,32,352):: GO
TO 260 ELSE 370
380 A=4 :: DIM FU(200):: GOS
UB 550 :: DISPLAY AT(4,10)SI
ZE(8):"UNSTETIG"
390 DISPLAY AT(10,1)SIZE(13)
:"Eingabe von H" :: ACCEPT A
T(10,15)VALIDATE(NUMERIC):H
:: IF H=0 THEN 390
400 DISPLAY AT(11,1)SIZE(13)
:"Eingabe von N" :: DISPLAY
AT(12,1):"N(N=2,4,6...)" ::
ACCEPT AT(11,15)VALIDATE(NUM
ERIC):N :: IF N=0 THEN 400
410 IF N/2<>INT(N/2)THEN 400
420 DISPLAY AT(13,1):"Eingab
e von"
430 FOR I=0 TO N :: DISPLAY
AT(14,1):"F(";I;")" :: B=LEN
```

```
(STR$(I)):: ACCEPT AT(14,7+B
)VALIDATE(NUMERIC):FU(I):: N
EXT I
440 DISPLAY AT(16,1):"Moecht
en Sie Funktionswerte aender
n (J oder N)" :: CALL KEY(3,
K,S):: IF K=78 THEN 470 ELSE
 IF K=74 THEN 450 ELSE 440
450 CALL HCHAR(16,1,32,64)::
 DISPLAY AT(16,1)SIZE(7):"WE
LCHE F(x)?" :: ACCEPT AT(16,
9):C
460 DISPLAY AT(14,1):"F(";C;
")" :: B=LEN(STR$(C)):: ACCE
PT AT(14,7+B)VALIDATE(DIGIT)
:FU(C):: GOTO 440
470 ER=0 :: FOR I=2 TO N STE
P 2
480 ER=ER+4*FU(I-1)+2*FU(I)
490 NEXT I
500 ER=ER-FU(N)*2 :: ER=(ER+
FU(0)+FU(N))*H/3
510 DISPLAY AT(19,1):"L=";ER
520 DISPLAY AT(24,1):"Weiter
 mit REDO oder BACK" :: CALL
 KEY(0,K,S):: IF K=15 THEN 1
40 ELSE IF K=6 THEN 530 ELSE
 520
530 CALL HCHAR(10,1,32,480):
: GOTO 390
540 GOTO 540
550 DISPLAY AT(1,2)ERASE ALL
:"{!!!!!!!!!!!!!!!!!!!!!!!!!;
" :: CALL VCHAR(2,4,36,A)::
CALL VCHAR(2,29,37,A)
560 DISPLAY AT(2+A,2):"}####
#####################~" :: DI
SPLAY AT(3,4)SIZE(22):"SIMPS
ON'SCHE NAEHERUNG" :: RETURN
```

BÖRSE

TI 99/4A Exbasic 100,–, Statisticm. 30,–, Buchhaltungsmodul 50,–, Parsec 20,–, Diskmanag. Modul 20,–, TI-Revue-Cass. 2 + 3/84 u. 7 + 8/85 u. SI-Special 2 Cass + 3 Cass voller Spiele 20,–, Datenverw. + Analyse 40,– DM. Suche TI-Artist, Multiplan + TI Logo. Tel: 06182/26186

Verk. wegen Aufgabe ca. 100 Top-Progr. für 70,– Ext. und TI-Basic, sowie 60 Spitzenprogr. für 40,– DM (Flugsim., 3D-Tennis, Sportspiele etc.) u. orig. Schlüter-Software auf Cass. Tel: 02541/4153 nach 15 Uhr

---

High Freaks!
Wer hat Lust mit mir E/A-Progr. zu tauschen??? Liste

Biete TI 99/4A + Cass.rec. m. Verbindungskabel + Ext. Basic + Lerncass. TI Basic + versch. Sachbücher + zahlr. Spielmod. Tel: 0228/281000, ab 14.30 Uhr

---

KABOOMMM
Verk. dt. u. engl. Bücher für den Texas! VB. 25,–! Topangebot: Irvel, Irvel 2, E.T., Surgeon + Batman vom Autor! 10,– DM! Weitere Infos bei: Manfred Lipowski, In der Wann 165, 462 Castrop-Rauxel, Tel: 02305/72237, Mo - Fr. 18 - 22 Uhr, Sa, So, ab 10 Uhr.

TI 99 Anwenderprogr. in Ex-Basic + Tape z. B. Mathsassistant (über 40 Rechenoperationen) o. Masterplaner (Tabellenkalk.) uvm. Adresse: Dirk Junghans, Am Fort-Biehler 9, 6503 MZ-Kastel

---

TI 99/4A + Box + 32 K + Ex Basic + E/A + Grafik-Tabl. + TI-Writer + 3D-World + TI-Artists + Kull + Tennis + The Mine + ID-Data/ID-Kto. + ca. 10 Module (adv. Return To Pirat, Fanthom ...) VB 2000,–. Tel: 0231/486516, ab 16 Uhr

---

Verk. TI-Writer 80,–, Ed./As. 70,–, Terminal-Emulator 50,–, Disk-Manager 40,– DM Rainald Prinzensing, Geitlingstr. 27, 4630 Bochum 6, Tel: 02327/61592

Verk. TI 99/4A (140) + Ext. Basic (170) + TI-Writer (100) + Schach (80) + Adven. (70) + Mini Mem. (110) + Disk Manager (60) + Ed./Ass. (60) + Parsec (60) + Statistik (40) zusammen für 750,– DM od. einzeln (Preise siehe Klammern). Ralf Schmitz, Tel: 02174/40654

KABOOMMM
Verk. Module! Z.B. Pole-Position, Congo-Bongo, Fathom 40,–! Car Wars, Wumpus u.s.w. 20,–! TE-2 Modul 60,– u.s.w.!!! Mehr Info über weitere Module (24 insges.) gibt es bei: Manfred Lipowski, In der Wanne 165, 462 Castrop-Rauxel 4, Tel: 02305/72237. Auch Tausch geg. gleichwert. Module!

Verk. folgende TI-Module: Schach 60,–, Othello 30,–, Blackjack 30,–, Pers. Rec. Keeping 50,– DM. Mini Memory VB! MBI-Interface 150,– DM. Suche Tauschpartner für E/A-Programme. Frank Müller, Ursfelderstr. 49 5014 Kerpen-Türnich, Tel: 02237/8276

Verk. weg. Systemwechs.: E/A-Paket 80,–, 32 K ext. m. Load-Interrupt-Taster 100,–, ext. Disk-Contr. DSDD u. Laufw. (Atronic) 700,–, Joysticks 20,– DM. Bei: Torsten Beuck, Horner Redder 14, 2000 Hamburg 74

Verk. TI + Ex-Bas., Joyst., Speech, MBI-Centr., 8-f-Exp, Cass-Kab, Module: Sp.-Edit., Dat. + Analyse, Text + Dat., FiBu, Invad., Tombst., Mus.-Make, Oth., Pars., Micro-Surg., Moonsw., Jgl.-Hunt, Dig Dug, Fußb., Munch m., Dong Kong, Bücher Tips + Tricks etc., Bas. Kurs, GP 550 Printer. Meschkat, Pf. 1330, 3502 Vellmar

---

TI 99/4A: 2 Konsolen, Ex-Basic, P-Box (defekt), RS 232-Karte, Disk-Drive, 32 K-Karte, Editor-Assembler, TI-Writer, Joystick Brother CE-60 m. Interface, Multiplan u. 7 Spielmod. auch einzeln zu verkaufen. Tel: 069/504575

Suche ext. 90 K-Laufw. f. orig. TI-Contr. (mögl. m. eig. Stromvers.) sowie Sprachsynt. Angebote an: Hauke Hölterling, Carl-Schutz-Str. 15, 6070 Langen. Tel: 06103/72518

Tips & Tricks für TI 99/4A, Teile I + II, jeweils 1 - 3, 5 KB (Ex-Basic). Progr. auf Kass. geg. Einsendung von 10,– DM. Peter Hielscher, Am Wall 22, 4401 Saerbeck

---

TI 99/4A Konsole m. eingeb. 32 K-CMOS Erw. zu verk. Rechner ist z.B. in Assembl. 2 x so schnell wie mit normalen 32 K, da alle 16 Datenleitungen direkt am TMS 9900 angeschl. sind. Nähere Info: Tel: 06257/83247

TI 99/4A Box Floppy m. Karte Ex-Modul RS 232 + Karte. 32 K Multiplan TE II Diskmanager, Drucker GP-100 Zenith-Monit. Joystick, Dateiverwaltung, 3 D Welt. Alle Handbücher. Div. Kabel. Preis VB. Alles auch einzeln. Hoffmann, Tel: 07022/46542, ab 12 Uhr

---

TI 99/4A, Ext. Basic, Siemens-Rekorder mit Kabel, Video-Chess, ca. 30 Zeitschr. 5 Bücher, 2 Joysticks, Progr., Superpr. Kompl. f. 450,– DM Tel: 09261/20790, nachmitt.

Zu verk. TI 99/4A m. Joysticks. Module: Datenverw. u. Analyse, Statistik, Personal-Generator, Kass: Haushaltsprogr. Div. Literatur, VB: 400,– DM. Tel: 0461/51509

Verk. TI-Module: Soccer 25,–, TI-Invader 30,–, Alpiner 30,–, Mash 40,–, Speech Editor 50,–. Suche: Editor/Assembler. Peter Glöde, Dorfstr. 43, 2362 Wahlstedt

---

Suche dringend einen Sprachsynthes. f. TI 99/4A. Mögl. m. Anleit.! Bezahle für das Gerät in 1A-Zustand + Versand u. NN max. 90,– DM. Bitte melden bei: Andreas Scholz, Londoner Ring 6, 6700 Ludwigshafen, Tel: 0621/666424

TI-Floppy aus Exp.-Box Shugart 400L m. kompl. Service-Unterlagen VB 150,– DM. Tel: 0711/3718 29

BÖRSE

Verk. TI 99/4A m. viel Zubehör z.B. Speichererw. Ex. Basic 2 Plus. Tel: 0211/422216

Verk. TI 99/4A + Pac Man + Kleingrafiken + 92 Spiele + TI-Fachzeitschr. + Rec.-Kabel + Infocass. + Handbuch + Anleitung f. VB 250,– DM. Tel: 08142/52873, ab 15.00 Uhr

TI 99/4A: Ext. Doppellaufw. i. Box + ext. Disc. (Texas) + Diskmanager 680,–; RS 232 (Atronic) 2 Ports ext. 180,–; 32 K + Centronic (Atronic) ext. 160,–; Tips + Tricks 20,–; Spielen, lernen, arbeiten 20,–. Chipprogr. 7,– 77 Basicprogr. Lon Poole 15,– DM. Tel: 06182/26186, Harald Ruppert

Suche das Adventuremodul. Erwin Kinslechner, Eibesbrunnergasse 1/10, A-1120 Wien

Verk. Pers. Fin, Aids + Bas.-Progr. Rout + Market simulation + Oldies II + Hardcopyroutinen, je 20,– DM. Tel: 08708/759, Semm

Verk. Mini Mem., Editor Ass. Mulitplan, je 150,– DM, Tex-Forth 60,– DM. Tel: 08708/759, Semm

Verk. TI 99/4A + Ext. Basic + dt. Handbuch + Tips + Tricks + Rec.-Kabel 250,– DM. Tel: 05371/16150, ab 18.00 Uhr

Editor Ass. Buch 30,–; Modulator, Ch 36 + Fbas 40,–; Tastatur 20,–. 5 x Spiel Modulen, Parsec usw. 150,– DM. Tel: 06155/4692

!Zu Verschenken!
habe ich nichts. Aber fast: TI 99/4A kompl. System. Liste geg. Freiumschl. J. Müller, Espanstr. 84, 8510 Fürth

Hallo TI-User!
Wenn Ihr Euch einem Userclub anschließen wollt, dann denkt an REX-SOFT! Wir versorgen Euch u.a. m, Ass.-Freeware. Eine Info gibt es gegen Rückporto bei: Rex-Soft, Pestalozzistr. 7, 2090 Winsen/Luhe

Verk. Universaldrucker Cosmos JP 80. Voll grafikfähig, Aufl. 640/1280 Punkte/Zeile und voll Epson kompatibel, 9 Nadeln vertikal, 30 Mio. Zeichen, VB 500,– DM. Tel: 0202/783901, ab 18 Uhr

Achtung TI-Freaks!!!
Verk. TI 99/4A + Ex. Basic + dt. Handbuch + Data Rec. f. 450,– DM. Tel: 07247/22542, nach 18.00 Uhr

Verk. TI 99/4A mit eingeb. 32K-Byte Erw. + Ex-Basic Modul f. 400,– DM, Tel: 06201/75170, Kohout

Verk. TI 99/4A Konsole, 3 J. alt + Software + Rec. Kabel + Bücher, VB 200,– DM. Tel: 02761/61749

Suche: ROM + GROM d. RS232 a. Cass. gespeich.! H.P. Thelen, Sebastianstr. 165, 5300 Bonn 1, Tel: 0228/628907

Verk. TI 99/4A f. 100,– DM sowie versch. TI-Rev.-Hefte zum halben Preis. Tel: 09871/9835

Bernstein-Monitor m. Kabel f. TI 99/4A f. 300,– DM zu verk. Johann Schmitz, Pf. 900 771, 5000 Köln 90

Verk. günst. meine Comp.-Magazinsammlung. Bitte Liste unter Beilage des Rückportos anfordern! Johann Schmitz, Pf. 900 771, 5000 Köln 90

TI 99/4A, Ex-Bas, Parsec, Schachmod., TI-Rec., Joyst., Cass.- Softw., Literatur! Billig! Tel: 06192/28956

Verk. TI 99/4A Anlage. Voll ausgeb. Einzeln o. zusammen. Info: Tel: 02331/586672

Tausche Software f. Ed.-Ass. Modul geg. TE II Modul u. Adv. f. Adv. Modul. P. Mertineit, In der Olk 20, 5501 Gusterath

Verk. Progr. f. E/A u. Ex-B. a. Disc. o. Cass. Listen geg. Rückp. P. Merineit, In der Olk 20, 5501 Gusterath

Weinheimer aufgepaßt! Besitze ausgeb. TI. Suche Kontakte & tausche Software!!! Meldet Euch bei: Oliver Arnold, Hauptstr. 44, 6946 Gorxheimertal, Tel: 06201/21429

Verk. 80-Zeichen-Drucker Ti-Silent 743 m. RS 232-Anschlußkabel (orig. TI) u. Papier 250,–. Original TI-RS232 Schnittstelle (ext.), 32K-Erw. eingebaut! 500,– DM. M. Becker, Tel: 06201/16272

Verk. Orig. TI-Diskettanlaufw. 90 K-RAM f. TI-Box. 360,– DM. Tel: 0221/843625

TI 99/4A 150,–, dazu Ex-Basic, nicht einzeln! 150,– DM, TI-Intern 25,– Apple kompl. System 2000,– DM. Tel: 089/6093425

Resetknopf, Zehnertastatur, Rabatte für Mitglieder, Clubheft, Prg.Speicher, USA-Service, 32 K in Konsole, Drukkerkabel uvm. im TI-CLUB BAUNATAL, Matthias Orf, Birkenallee 34, D-3407 Baunatal 1, Tel: 0561/497990. Info 50 Pfg. od. aktuelles Clubheft 3,– DM anfordern ... es lohnt sich!

Wenn die 256K-Speichererweiterung lieferbar ist, erscheint im Clubheft des TI-CLUB BAUNATAL der ausführlichste Testbericht dieser Erweiterung. Wollen Sie den Bericht lesen? Ja, dann fordern Sie noch heute geg. 3,– DM Clubhefte an – bei: TI-CLUB BAUNATAL, M. Orf, Birkenallee 34, D-3407 Baunatal 1, Tel: 0561/497990

Orig. TI-Erweiterungsbox, 32K, Disc-Kontroller, 1 Int. und ein ext. Laufwerk, RS 232-Karte, Buchhaltungsmodul, Personal Rec. Keeping + Maker, Schaltunterlagen, versch. Literatur, sehr viel Software auf Disk., Mathematik 1, Basic Progr. Routinen II u. III, Baustatistik, kompl. abzugeb. für 1400,– DM. Günter Trunk, Urnenstr. 9, 6700 Ludwigshafen 25, Tel: 0621/678578, ab 17.00 Uhr

Verk.: TI99/4A + Kass.-Kabel + orig. X-Basic + TI-Joysticks + Spiele + Bücher + 3 Spielmodule zu TI (Parsec + Schach + Munch Man) + 9 TI-Revue + 1 TI-Special. Nur zusammen, Preis: 520,– DM. Tel: 05064/6177

Verk. 1 TI Rec.Kabel 10,–, 1 Netzteil vom Vorgänger des TI 99/4A (5V/1.6A, + 12V/o.5A, -5V/o.15A), Preis VB, Ersatzteile vom TI 99/4A, Leercass. C 60 a –,50 DM.
Suche: Module: Miner 2049, Burgertime, Popeye, Video Chess, Jungle Hunt, Defender, Dig Dug, Pole Position u.a.
Friedrich Haage, Rudolfstr. 9, 7460 Balingen Fr.

Verk.:
Konsole TI 99/4A, s/w Monitor, Speech Synthesizer, TI-Box m. Flex Cable Karte, 32 KB Karte, RS232 Schnittstellen Karte (1xParallel, 2 x seriell), Disk-Controller, 5 1/4 Disklaufw. (alles orig. TI)
Modulsoftware: Diskmanager II, Ex. Basic, Ed.-Ass., Multiplikation I, Add. u. Subtr., Minus Miss., Parsec, Alpiner (alles orig. TI)
Literatur: 99 Spezial I, Ed. Assembl. Handbuch (dt. u. engl.), Mini Memory Handbuch, TI-Rev. (15 Hefte), Computer Contact (7 Hefte)
Ca. 10 Disk. m. Software, Kass.Rec.Kabel (doppelt), Joystick, Druckerkabel (parallel), Schaltpläne. Preis– VB. Neuw. ca. 3500,– DM, Tel: 06403/2289, ab 16 Uhr

2K-Speicher für nur 100,– direkt ansteckbar zu best. bei: C. Mohr, Friesenweg 38, 4133 Neuk.-Vluyn, Tel: 02845/21507

32K-Erw. z. Einbau i.d. Konsole (Superschnell, da direkt. Anschluß aller 16 Datenleit.) Lötarbeit erforderl. Verk. auch einige Module sowie Buch 99 Special. Preis: VB! Tel: 06257/83247

Neuw. Software zu verk.! Module: Dig Dug 15,–, Return to Pirate Isle 25,–, Parsec 20,–, Munch Man 20,–, TI Invaders 20,–, Chisholm Trail 10,–, Alpiner 20,–, Adv. + allen 11 Cass. 75,–, 70 Cass. m. ca. 400 Progr., tägl. 15 - 17 Uhr Volker Niemeyer, Tel: 04202/81279

Suche TI-User zwecks Erfahrungsaustausch im Raum Düren. Stefan Bachem, Arnoldsweilerstr. 52, 5160 Düren, Tel: 02421/17673

Schnellste Dateiverw. m. Kass. Ex-Basic-Progr. auf Kass. geg. Einsendung von 10,– DM. Peter Hielscher, Am Wall 22, 4401 Saerbeck

Endlich ist es da!!! Dt. Handbuch für TI-Writer u. TI-Forth. Preis pro Buch: 45,– DM. Ralph Weber, Langemarckstr. 54, 8906 Gersthofen

Achtung! Tausche Mini Mem. orig. TI + Literatur geg. Sprachsynthes. M. Macke, Tel: 0431/526083

BÖRSE

Verk. Auswahl aus meiner Progr.Sammlung (TI u. Ex-Basic). Eine C-60 Kass. voller Progr. für nur 10,– DM. Interessenten bitte melden bei: Andreas Scholz, Londoner Ring 6, 6700 Ludwigshafen

---

Disk-Controller und TI-Laufw. intern für P-Box zu verk. Preis VB! Tel: 062 57/83247

Für alle die keine Schnittstellenkarte haben. Ich biete für nur 150,– DM das MBI-Interface TI auf Centronics an. Nur 1 x benutzt. Anfragen bitte an: Paolo Pririllo, Battonnstr. 30, 6000 Frankfurt/Main 1

Die Sensation überhaupt! Das neueste Spiel von MG Soft in Ex-Basic: Borondo! Zu einem Sensationspreis: Statt 20,– nur 10,– DM (einmaliger Preis). Also 10,– DM-Schein in Briefmarken an: M. Görgen, Helgolandring 122, 4300 Essen 1

Verk. TI 99/4A + Ex-Basic, + Joystick + Rec. Kabel + Rec. + Mod. (Alpiner, TXT u. Dateiverw. + Soccer) + Bas. f. Anf. + Bücher + 3 Aktenordner. Listings + Ifnos + Progr. Cass. nur kompl. 630,– DM. H. J. Beckmann, Untere Dorfstr. 165, 5900 Siegen

---

Achtung!
TI 99/4A-System (kons., Box, D-Laufw., Module, Disk. über 100 Hefte, Bücher, Joystick u.v.m.). Alles gepfl. u. neuw., umst.halber 40 % u. NP abzugeben. Kpl. Liste gegen frankierten Rückumschlag bei Jürgen Müller, Espanstr. 84, 8510 Fürth

---

Verk. Radix Miniassembl. inkl. 2 Software-Cass. für 100,– DM od. tausche geg. 3 gute Spielmodule. Angebote an: Werner Clausen, Mathildenstr. 17, 2390 Flensburg

---

Biete an: Die Module: Monn Patrol, Buck Rogers, je 30,–; Datenverw., Alien Add., Othello, je 25,–; Rec.-Kabel 12,–, 10 TI-Rev. 7,–, Good Oldies Plays I, II, 10,–; Speech Synth.: 85,–; 50 Progr. 20,– DM. Tel: 07621/84508!!!

Speichererw. 32 K, batteriegepuff., kontroll-LED, extrem durchgef. BUS neu für 190,– DM. Tel: 0241/86447.

High, Freaks! Wer hat Lust mit mir E/A-Progr. zu tauschen?!? Liste an: Ralph Benzinger, Wilhelmstr. 65, 6800 Mannheim 51

Neue Scott Adams Adv.
The Hulk, Spider-Man, Bukkaroo-Banzai, The Sorcerer of Clay morgue castle, Sott Adams Lösungsbuch für alle Adv. Melden bei: J. Huppert, Hohenstaufenstr. 27, 7141 Möglingen. Tel: 071 41/461147. The Big Boiler. Tausche Adv. + Ass.Progr., verk. RS 232 + Drucker Kabel.

XXXSoftwareXXX
Verk./tausche Basic u. Ex. Basic Spiele. Suche günstige Module: Jungle Hunt u. Burgertime u. auf Kass.: Der schwarze Kristall. Günter Simoner, Baslerg. 50-66 A-1232 Wien

---

TI 99/4A kompl. Anlage orig. Peri-Box (32 K, RS 232, Diskkontr., 1 LW), SpSynth, Modulexp., Epson RX80F/T + 26 Module (Ed/Ass, MM, TI-Writer, XB, Disk-Man etc.) 2 gef. Disk-Boxen, Kabel, Joystick, 5 kg dt. + engl. Literatur uvm. VB 2200,– DM Tel: 069/6701252

Verk. geg. Angebot TI 99/4A + Netzteil und Palmodulator, Tel: 0203/583718, Roland Kegel, Lehrerstr. 32, 4100 Duisburg 11

Suche im Raum Dortmund Kontakt zu anderen TI 99/4A-Besitzern. Melden bei: Andreas Berkenbrock, Evinger Berg 8, 4600 Dortmund 16

Verk. 32 K, durchgef. BUS' extern (nach TI-Rev. 9/85) Opt. 1 A, schw. Gehäuse, Kontroll led. FP: 170,– DM. Außerdem: 32 K, dto. ohne Geh. m. Anschluß, voll funktionsfähig 140,– DM. T. Külpmann, Lützowstr. 54 a, 5800 Hagen, Tel: 02331/21454

Verk. TI99/4A mit Peri.-Box, Laufw., Contr., Ext. Basic, TI-Writer, Ed.-Ass., Apesoft-Grafik, Pac-Man, Invaders, Alpiner, Software + Literatur kompl. o. einzeln! H. Rögner, Lammsgasse 12, 8500 Nürnberg 1

Fast geschenkt!!!
Org. TI-Joysticks abs. neuw. 15,– DM + Modul: Datenverw. u. Analyse 20,– DM! Tel: 02451/46608

Orig. RS 232, Druckerkabel, Zero-Zap, TI-Forth, TI Rechnungst., Div. Magazine von TI Soft Club Belgien, TI-Cass.-Basic Kurs, 4 neue Scott Adams Adv., 5 Spiele, Lösungsbuch für Adv.-Spiele, tausche Ass.-Progr. The Big Boiler, Tel: 07141/461147

Verk. Orig. US-Ext.-Basic m. Handbuch, neu 120,– DM, 32 KRAM ext. 150,– DM. Anfragen schriftl. an: Martin Sommer, Am Vogelherd 42, 5060 Berg. Gladbach 2

Superspiele für den TI 99! Über 300 Progr. in TI- und Ex-Basic für 2 - 8,– DM. Spielhallenhits, Advent., Action, Denk- und Glücksspiele, Weltraum- und Sportspiele. Info 1,– DM bei Frajo Fry, Sachsenstr. 30, 4350 Recklinghausen

---

Große Auswahl an günstigen Top-Spielen in TI- und X-Basic: Fußball, Hyper Olympic, Quasimodo, Ghostbusters, Manic Miner, Shuttle Command, Diablo und 300 weitere Spiele. Info 1,– DM bei Ute Simon, Sachsenstr. 30, 4350 Recklinghausen.

32K-Byte Karten für 110,– DM! Direkt an den TI ansteckbar! Jede an mich geschickte Postkarte bekommt 1 32K-Karte!!! Bez. per NN 4133 Neuk.-Vluyn, Friesenweg 38, W. Mohr!!!

Verk. f. TI/99/4A
Kodeknacker (EX,C,D) 5,–; Tape Directory (Ex, C,D) 5,–; Star Texter (EX, C,D) 7,–; Star Datei (EX, D,C) 8,–;Telefonregister (EX,C,D) 7,– DM. Bei Marco Wintzer, Mühlstr. 42, 6070 Langen

---

Restposten Diskettenlaufwerke
Slim 5-1/4" DSDD TEAC FD55B 160,–
GROMplatine mit Gehäuse (Spiel) 22,–
Module Statistics o. Othello je 20,–
Comerz. Zeichnen Programm z.B. für Schaltbilder o. Layout 9 Bildschirme auf 1 DIN A4 Blatt auf Disk. 30,–
E. L. Levits, Tel: (040) 299 46 09, Roggenkamp 3, 2000 Hamburg 60

Verk. Ext. Hardware: Eine RS232 (2 Ports) f. 250,–; ein Disklaufw. f. 280,– DM Orig. von TI. Tel: 08161/61537

TI 99/4A + Ex-Basic + Lit. zu verk. Preis 450,– DM/VB Tel: 0451/495678, 2401 Badendorf, R.H. Wilde

---

Ich tausche Progr. in TI u. ExB. Liste an: Erwin Kinslechner, Eibesbrunnerg. 1/10 A-1120 Wien

---

Verk. TI-Ext.Basic 150,–; TI-Revue 3/84 bis 6/86 (13 Stück)50,– + TI Special 1, 2 u. 4, 25,– DM. Peter Koch, Poststr. 57, 4755 Holzwickede, Tel: 02301/2052

!Hallo TI-Freaks!
Ich verk. Mini Memory sowie Spielmod. Tel: 0201/712816

TI 99/§A, Ex-B., Joystick-Adapt., massig Softw. + Literatur 290,– DM.
Tel: 0911/435909, öfters versuchen.

Suche günst. Disk.Contr. f. BASF-Laufwerk 6106 - Ext. Klaus Schildbach, Rektor-Roth-Str. 3, 6501 Nieder-Olm

---

Suche Kontakt zu anderen Forth programmierenden TI-Usern. P. Kliem, E.-Brandströmstr. 37, 5042 Erftstadt

---

Verk. Übersetzung E/A über 400 S + Bspl + Tabellen, wie im orig. manual f. 49,80 DM Andreas Pack, Eickelerstr. 60 4690 Herne 2

Verk. f. TI 99/4A ext. 32KRAM, Akku, gepuffert mit Progr. 230,– DM, neu. H.J. Rosin, Eicherdorffweg 50, 5308 Rheinbach

---

Diskeditor, Fast-Copy, GPL-Disassembler f. E/A, XB, MM u. 32 K. Alle sofort lieferbar: 100 % Masch.Spr. Info geg. Freiumschl. Mathias Eichhorn, Ziegelheck 1, 6240 Königstein 4

Verk. für TI/99/4A 12,–; Ghostbusters (D,EX, 32K) 9,–; Hover-Bower (C od. D, Ex) 7,–; Adventureland (D, Ex) 9,–; Land des Grauens (Ex,C) 7,–; Space Battle 1, 2, 3 (Ex,C,D) je 7,– DM. Bei Marxo Wintzer, Mühlstr. 42, 6070 Langen.

Super! Verk. TI/99/4A + Extended Basic + Handbücher dt. u. engl. + Joyst. + Rec.-Kabel + Kass. m. ca. 100 Supersp. + ca. 50 Listings zum Sensationspreis von 250,– DM. Roland Stahl, Schubertstr. 8, 8411 Duggendorf

BÖRSE

Ti-User: Laßt Euch von Geschäftemachern nicht ausnehmen! Bei uns gibt's US-Freeware gratis – wie sich's gehört! Nur dem Autor schickt Ihr Euer Geld – Kontakt: Michael Möller, Wehrhofstr. 10, 6000 Frankfurt 90, Rückporto f. Info beilegen.

Editor/Ass. 140,–, Mini Memory 130,–, Ex. Basic 150,–, Terminal-Emulator 50,–, Statistics German 20,–, TI-Invaders 30,–, Adventure + 2 Cass. 80,–, Parsec 30,–, Sprachsynthesizer 110,–, Joysticks 30,–, X-Box + Laufw. + Modul + Karte + Anleitung 1150,–, RS 232 Karte 300,–, TI 99/4A + 3 Bücher + Rec. Kabel + Anleitung 330,–, zusammen 2150,– DM. Bestellen per NN, D. Past, A-Roßhaupter-Str. 104, 8000 München 70

Biete/Suche/Tausche TI99/4A Hardware ext. tint. sowie orig. Software. G. Bürger, Im Pump 5, 3051 Anhagen, Tel: 05725/6409, ab 17 Uhr. Suche TI-Calc und dt. Handbuch

---

Drucker Gemini 10X zu verk. Preis: ca. 500,– DM, Tel: 06081/7477, N. Keller, Taunusstr. 1, 6392 Neu-Anspach

---

TI 99/4A, TI99/4A, TI99/4A Verk. RS 232 Schnittstelle für nur 150,– DM! Verk. viele Module (Buck Rogers usw.) von 10 - 40,– DM. Interessenten bitte an M. Schenk, Rosenweg 4, 7107 Bad Friedrichshall

---

TI 99/4A ist noch da! TI99/4A ist noch da!
Wer hat Lust mit mir Progr. zu tauschen oder Hardware (meine) zu kaufen? Einfach 80 Pfg.-Briefmarke an: Markus Schenk, Rosenweg 4, 7107 Bad Freidrichshall

---

O Mein Gott!
Gibt es denn im Umkreis von Rad-Hersfeld nur 2 jugendliche TI-Benutzer??? Wenn es tatsächlich mehr sind, dann bitte unter Tel: 06621/61416, ab 13.00 Uhr melden!

Verk. Spielmod.: Attack-Invaders u. Tomb.-City, je 20,– DM od. tausche geg. andere Spielmod. (Schach o. Advent. bevorzugt) Verk. auch Buch TI-Bas. - Exbas, VB 25,–, auch Tausch geg. Special + I o. II. Tel: 08122/13094

99'er MAILBOX
Erste deutsche Malbox für *den TI99/4A. TI-Box*
Online: Sa + So 19 - 24 Uhr
Para's: 7/1/Even
Tel: 07142/45129

MAILBOX für TI99/4A
Wo sind die texanischen Hacker?
Erste deutsche Mailbox für TI99/4A. *TI-Box*
Online: Sa + So 19 - 24 Uhr
Para's: 7/1/Even
Tel: 07142/45129

TI 99 wegen Aufgabe: 30 Module, EA, XB, Writer, Multiplan, DBase, Testmodul u.a., ca. 250 Disk m. Software, Stück 5,– DM, 40 Bücher und anderes mehr, u.a. orig. US-Produkte, Liste gratis: G. Lesnik, Am Jungbrunnen 9, 4600 Dortmund 18

Suche Diskkontroller, ext. Speichererw. und LW Schachmodul. E. Müller, Langenbochumer Str. 371, 4352 Herten, Tel: 0209/610829

Suche def. TI-Teile Box, Karten, Module usw. Angebote an: Walter Rossmanif, Tel: 06035/6931, ab 20 Uhr. Suche Grafikprogr./Hardkop, TI-Seikosha, GP 550A

Verk. f. den TI 99/4A folgende Module: Tunnels of Doom 50,–, Schach 50,–, Othello 25,–, Datenverw. 50,–, Buchungsjournal 100,–.
Folgende Kass.: Flugsimulator 30,–, Marketingplanspiel 20,–, Basic Routinen 1 40,–.
Folgende Bücher: Assembler-Handbuch für Mini Mem. in dt. 40,–, Chip-Buch, Progr. für TI 99/4A 10,–.
Folgende Hardware: P-Code-Karte mit Pascal Software (Compiler, linker, Assembler, Utilities) u. Handbuch 750,– DM. Norbert Röser, Goethestr. 31, 8750 Aschaffenburg

---

Verk TI 99/4A, 2 Konsolen, 2 Adapter, 4 Spielmod, Ex Basic, EdAss., P-Box, Multipl, Diskman., Drucker, RS 232 Listings, V24 Schnittstelle, s/w-Monitor, Cassrec. Zeitschr, 99 Special 1 + 2, Tips + Tricks, u.v.m., Preis VB 3200,– DM, NP 4500,– DM. Tel: 06253/6847

TI-Multiplan zu verk. 150,– DM od. Tausch geg. TI-Writer. Tel: 040/5522677

Verk. folgende TI-Module: Tunnels of Doom (Graf.Adv.) Adventuremodul, je Modul + Kass. 50,– DM. Adv. Kass.: Ghosttown, Adv.Land, je 40,– DM. Anschr: Helmut Reuther, Bellerweg 4, 5449 buch, Tel: 06762/6455

Suche: RS232 f. P.-Box, Modul, Editor/Assembler. Anton Raederstorff, Pf. 216, 4123 Allschwil 1, Schweiz, Tel: 061/635361

Verk. Cons + Pal Mod m. Monitoranschluß, 32K-Karte, RS232Karte, Diskcontr., Diskdr., Per-Box, Xbasic, Ti-Writer, Modulexp, Preis 150,–, 150,–, 190,–, 200,–, 200,– 200,–, 100,–, 150,– 80,– und Einiges mehr. R. Wosse, Randenborgweg 125, Nieuwstadt (NL), Tel: 0031/4498/53598

Verk. P.-Box, 32 K, RS 232, Controller, Laufw. alles Ori. f. 3 Laufw. vorber.
P-Box, wie vor, jed. mit 2 TEAC-Laufw., beide Boxen techn. und opt. o.k.
1 99/4A, ladenneu, 3 dto. wenig benutzt. Rarität: 99/4 m. NTSC-Monitor (Grundig, voll o.k.
! RGB-Modul, 1 Modulexp., 3 Sprachsynt., 1 dto, neu, 1 Queme-Terminal/80 Zeich., f. TI-pass., m. Softw. dazu.
Module: 2 mech. Exb. Plus II, 3 orig. TI-Exbas, 1 dto, neu, 1 Mini Mem., 3 neue Buchungsjourn., 1 TE 11, 1 Speeched., 1 Lagerverw./disk., 1 Demo-Verk.mod., 1 Fitness-Training, 2 Hangman, 1 Video-Game I, 1 Wumps, 2 Schachmeister, 1 Tunnels of Doom, 1 Munch Man, 3 Tombstone City, 1 Driving Demon, 1 Chisholm Trail, 1 TI-Invaders, 2 Parsec, 1 Wewermania, 1 Bigf.
1 MB-Spracherkennungsyst., inkl. Baseball-Modul (dazu pass.: Severmania und Bigfoot-Module).
1 Orig. KULL-Compiler. Amerik. 99er Magazine. Sehr viele Progr. a. Disk. TI-Revue. Tel: 0441/54423, ab 18 Uhr.

Wir suchen TI-Fans i. d. ganz. Welt!!! Erwin Kinslechner, Eibesbrunnerg. 1/10, A-1120 Wien

---

Hallo TI-Freunde
In Heft 6/86 war meine Anzeige auf Seite 62, 4. Spalte. Leider habe ich die Anschr. vergessen. Hier alles nachträglich: Orhan Kiran, Hauptstr. 52, 7601 Schutterwalt, Tel: 0781/51884

Module zu verk. Multiplan, Burgertime, Congo Bongo, Personal Record Keeping, Preise ? DM. Frank Gindullis, Drosselkamp 12, 2200 Elmshorn, Tel: 04121/24243. Auch Tausch gegen Speech Synthesizer oder Modulexpander

---

Verk. wenig gebraucht: Matrixdrucker Seikosha GP 100 A, 80 Zeichen/Zeile m. Hardcopy-Progr. f. 250,– DM. N. Burkhard, Ringstr. 21, 6530 Bingen 17.

Zu verk. Handb. z. Advertizer Modul 20,– Fr. Brief-Vorauskasse o. NN. B. Schönauer, Alpenstr. 32, CH-2540 Grenchen.
Suche Eprombrenner + Software

Verk. Munch Man 25,–, TI-Invaders 35,–, Mind Challenger 30,–, Datenverw. + Analyse + Pers. Rep. Gen, je 35,–, Basic Progr. Routinen 2 + 3, je 50,–, Tunnels of Doom 50,–, ID-Data 2 50,–, Torpedo Basic 50,–, Hagera Ass-Kurs 40,– DM. Tel: 08708/759, Semm

---

Verk. Mechatronic Extendet Basic + div. Bücher (Tips + Tricks etc.) + TI-Rev. 1/84, 3/84 - 6/86 + TI-Special (1/2) + 100 Progr. (Flugsim. etc.) VB 200,– DM. Manfred Kersen, Tel: 02591/6018

Thermo-Transfer-Drucker HR5, RS 232, druckt auf Normal- und Thermopapier, für Batterie u. Netz.
Inkl. Netzteil, Farbbändern und Kabel im Orig. Karton: 200,– DM. Tel: 06821/79825, Lucas, nach 17 Uhr

---

Achtung! Achtung! Achtung!
Die erste 99'er Mailbox – Logg doch mal rein
Online: Samstag von 19 - 24 Uhr
Sonntag von 19 - 24 Uhr
Tel: 07142/45129.
Bitte nur während der Online-Zeiten anrufen!

TI-Konsole defekt 80,–, Ext Modul 90,–, Handb. ex. engl. 10,–, Tip + Tricks, Data Becker 10,–, TI-Konsole z. Ausschlachten 50,– DM. Tel: 07141/482666

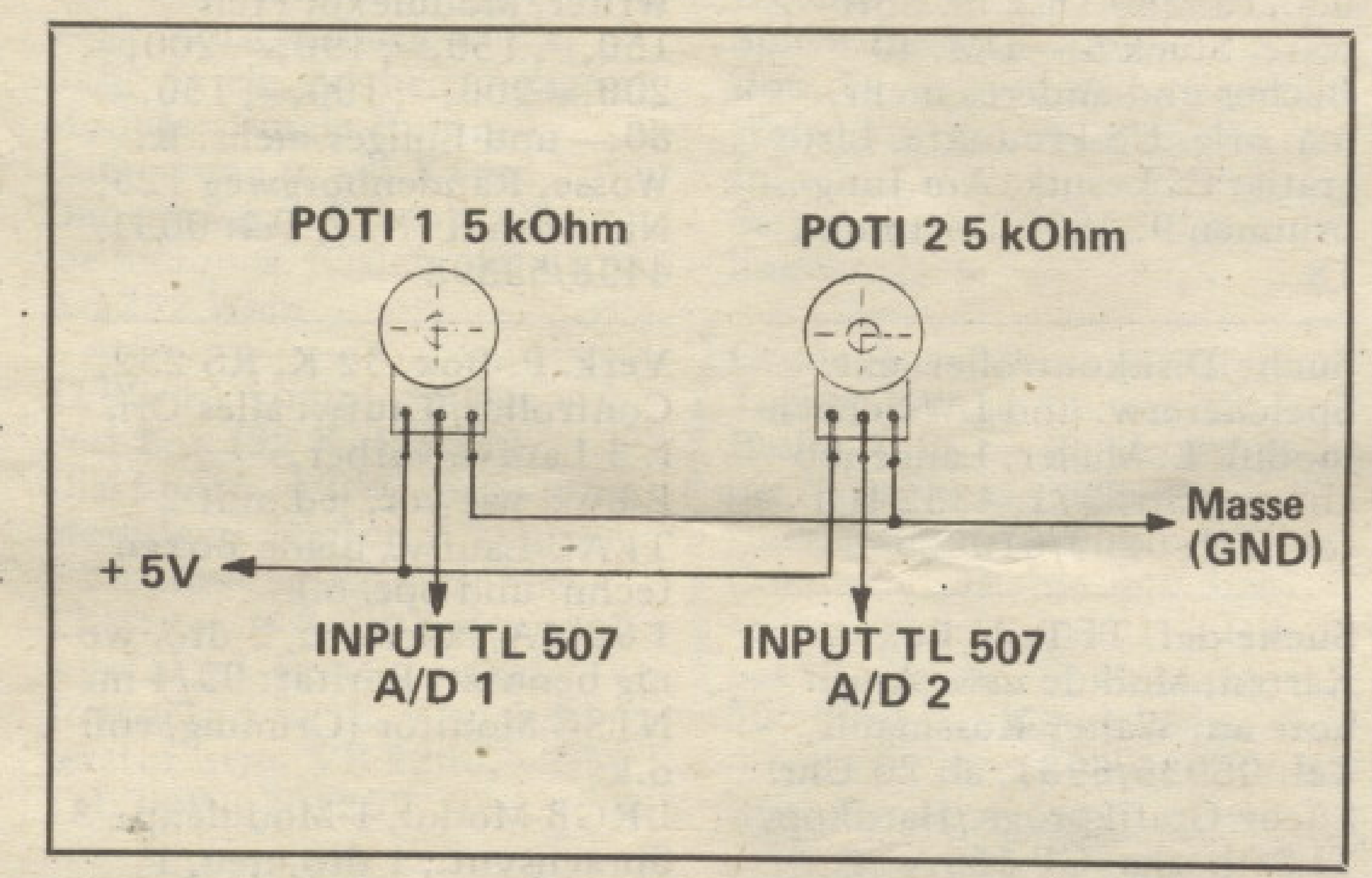
Dateiverwalt.Progr. superschnell in TI-Basic, auch in Kass.Version senden Sie nur 20,– DM in einem Umschlag mit Absender an: G. Schmidt, Reumotstr. 52, 4790 Paderborn

PROPORTIONALE JOYSTICK-STEUERUNG FÜR IHREN TI 99/4A

"Projoy" erlaubt die proportionale Steuerung von Sprites und basiert im Wesentlichen auf dem in TI-REVUE 3/86 vorgestellten A/D-Wandler.
Um eine proportionale Steuerung zu realisieren, braucht man eine Eingabe-Einheit. Die im Handel erhältliche 'TI-Maus' ist jedoch recht teuer und erfordert ein Diskettenlaufwerk.
Als preiswerte Alternative bot sich da ein Knüppel für Fernsteuerungen an, der bei Völkner in Braunschweig schon für 36,– DM zu bekommen ist.
Nun braucht man nur noch 2 der schon erwähnten A/D-Wandler sowie das I/O Modul aus TI-REVUE 2/84.

FUNKTIONS-BESCHREIBUNG:

Die von den Potentiometern des Knüppels kommende Meß-Spannung wird von den A/D-Wandlern digitalisiert und durch das I/O Modul in den Rechner eingegeben. Per Software können nun Sprites proportional zu den Knüppelbewegungen über den Bildschirm huschen – eine Hilfe z.B. für Grafik- oder Spielprogramme.



HARDWARE:

A/D-Wandler "1" anschließen, wie es in TI-REVUE 3/86 beschrieben wurde.
A/D-Wandler "2" wird folgendermaßen angeschlossen:

1. Pin 4/TL 507 (out) an Pin 2/74LS251(I2)
2. Pin 2/TL 507 an Pin 6/74LS259(03)
3. Pin 8/TL 507 (RES) an Pin 4/74LS259(01)
4. Spannungsversorgung herstellen
5. Anschluß des Steuerknüppels:

SOFTWARE:

Das Programm aus TI-REVUE 3/86 wurde auf 2 A/D-Wandlern erweitert. Um eine schnelle Ansprache der Sprites auf die Knüppel-Bewegungen zu gewährleisten, werden die Sprites durch das Maschinenprogramm positioniert. Durch Drücken einer beliebigen Taste wird ins Basic zurückgesprungen.

Format:
CALL LINK ("PROJOY", #SN,X,Y,K)
#SN = Spritenummer (1–28)
X = Reihe
Y = Spalte
K = ASCII-Code der gedrückten Taste.

Benötigte Geräte:
TI-Konsole
Ext. Basic
32 kB Erweiterung
I/O Modul
2 A/D Module

Kay Sievert

DIE CLUB-ECKE

Die Firma Texas Instruments gab vor einiger Zeit ein 'Care Package', eine Sammlung von Programmen und Informationen, heraus. Diese Sammlung sollten alle TI Userclubs bzw. User kostenlos, bzw. gegen Portorückerstattung, erhalten. Daß dies nicht immer der Fall zu sein scheint, zeigt folgender Brief:

Liebe TI-Redaktion!
Ich hoffe, daß ich mit diesem Brief anderen TI-Usern, was uns, drei TI-Freunden in Frankfurt, passierte, ersparen kann. Obwohl sich die Leute, die uns die Programme 'verhökert' haben, in der letzten Zeit in so mancher Veröffentlichung als eine Art 'Rotes Kreuz' für den TI-User darstellen, passierte folgendes.
Mein Freund fuhr mit dem Zug nach Bonn und traf den Herrn. Am Telefon war die Rede davon, das Care Package von Texas Instruments zu bekommen (von Geld war keine Rede – ist ja Freeware). Dann kam der Hammer. Pro Paket-Teil oder Programm 25,– Mark cash auf die Hand. Auch andere Kopien (Infocom) wurden zu diesem Preis angeboten.
Auf die Frage, wofür das Geld sei, wurde gesagt – wörtlich: „Ein Ausgleich für die lange Zeit des Wartens!"
Ein echter Gipfel der Unverschämtheit!
So beschlossen wir, einen Club o.ä. aufzumachen, die Freeware und das Care Package an jeden Interessenten gratis – wie es doch wohl richtig ist, abzugeben.
Von dem Geld hat der Autor, der sich die Mühe machte, bestimmt nichts gesehen und der hat's ja wohl als einziger verdient. Wir haben uns schwer geärgert.
Der 'Herr' handelte übrigens nicht auf eigene Kasse, sondern sagte, es sei für seine Gruppe, trat also offiziell auf!
Michael Möller
Wehrhoftstraße 10
6000 Frankfurt/Main 90

Wenn andere ebenfalls die Erfahrung mit einem solchen Kontakt machten, so möchten sie dieses auch an uns schreiben. Clubs, die dieses Care Package noch nicht haben und es, nachdem sie es erhalten haben, an andere TI-User kostenlos oder unter Portorückerstattung bzw. Einsendung einer Diskette, weitergeben wollen, können uns schreiben.
Heiner Martin

LISTING FÜR DIE JOYSTICK-STEUERUNG

```
*       '     ** PROJOY **
        DEF   PROJOY
CR      EQU   >1200
MW      BSS   32
BX      BSS   2
BY      BSS   2
SA      BSS   2
SN      BSS   2
KE      BSS   2
BS      BSS   2
DA      DATA  10
I1      MOV   R11,§SA
        LI    R12,CR
        SETO  R6
        LI    R5,127
        SBO   0
        SBO   1
        SBZ   0
L1      TB    1
        JNE   F1
        SBZ   1
        CLR   R6
        SBO   1
        DEC   R5
        JH    L1
        LI    R5,>7F00
        JMP   I2
F1      MOV   R6,R6
        JEQ   I2
        LI    R5,>FF00
OT      MOV   R5,§BX
        MOV   R7,§BY
        MOV   §SA,R11
        RT
I2      SETO  R6
        LI    R7,127
        SBO   0
        SBO   2
        SBZ   0
L2      TB    2
        JNE   F2
        SBZ   2
        CLR   R6
        SBO   2
        DEC   R7
        JH    L2
        LI    R7,>7F00
        JMP   OT
F2      MOV   R6,R6
        JEQ   OT
        LI    R7,>FF00
        JMP   OT
LO      MOV   R11,§SA
        LI    R0,36
        S     R0,§BX
        S     R0,§BY
        MOV   §BY,R1
        LI    R0,34
        MPY   R0,R1
        CLR   R1
        DIV   §DA,R1
        LI    R0,198
        S     R1,R0
        MOV   R0,R4
        MOV   §BX,R1
        LI    R0,44
        MPY   R0,R1
        CLR   R1
        DIV   §DA,R1
        LI    R0,262
        S     R1,R0
        MOV   R0,R5
        CI    R4,1
        JLT   O2
        CI    R4,192
        JGT   O2
        CI    R5,1
        JLT   O2
        CI    R5,255
        JGT   O2
        MOV   R5,§BX
        MOV   R4,§BY
        MOV   §SN,R0
        MOV   R4,R1
        SWPB  R1
        BLWP  §VSBW
        INC   R0
        MOV   R5,R1
        SWPB  R1
        BLWP  §VSBW
O2      MOV   §SA,R11
        RT
BR      MOV   §BY,§FAC
        BLWP  §XMLLNK
        DATA  >0020
        CLR   R0
        LI    R1,2
        BLWP  §NUMASG
        MOV   §BX,§FAC
        BLWP  §XMLLNK
        DATA  >0020
        CLR   R0
        LI    R1,3
        BLWP  §NUMASG
        MOV   §KE,§FAC
        BLWP  §XMLLNK
        DATA  >0020
        CLR   R0
        LI    R1,4
        BLWP  §NUMASG
        CLR   §>837C
        MOV   §BS,R11
        LWPI  >83E0
        RT
E1      LI    R0,>0200
        BLWP  §ERR
E2      LI    R0,>1E00
        BLWP  §ERR
*       '     HAUPT-PROGRAMM
PROJOY  LWPI  MW
        MOV   R11,§BS
        CLR   R0
        LI    R1,1
        BLWP  §NUMREF
        CLR   §>8354
        BLWP  §XMLLNK
        DATA  >12B8
        MOVB  §>8354,R0
        JNE   E1
        MOV   §FAC,R0
        CI    R0,1
        JLT   E2
        CI    R0,28
        JGT   E2
        DEC   R0
        LI    R1,4
        MPY   R0,R1
        AI    R2,>0300
        MOV   R2,§SN
TS      BL    §I1
        BL    §LO
        CLR   §>8374
        LI    R6,>2000
        BLWP  §KSCAN
        MOVB  §>837C,R1
        COC   R6,R1
        JNE   TS
        CLR   §>837C
        MOV   §>8375,§KE
        B     §BR
        END
```